SCIENTIFIC REPORT NO. 2

ON THE REPRESENTATION OF LIMITED INFORMATION BY MEANS OF PICTURES, TREES, AND ENGLISH-LIKE SENTENCES

by

Manfred Kochen
University of Michigan
Consultant to RCA Laboratories
Princeton, New Jersey 08540

Prepared for the Air Force Office of Scientific Research of the Office of Aerospace Research under Contract No. AF 49(638)-1148.

PREFACE

Most of the work described here was done in the summer of 1965 as part of our continuing research in the general area of machine problem solving. In the fall of 1965, a (limited distribution) report covering this work was issued at RCA Laboratories and at the Mental Health Research Institute of the University of Michigan. Plans to revise this report and to combine it with subsequent results of our work on representations in question-answering systems have delayed its wider distribution. However, since the work described in the report is relevant to much current research on graphic languages and question-answering, we have decided to extend its availability to the technical community in its present form, and to issue it as a technical report.

Saul Amarel
Princeton, N. J.
May 1968

ABSTRACT

In this paper we discuss means of representing states of the world which are easily described as pictures of triangles, circles, and squares in horizontal, vertical, or enclosure relationships; our study is oriented to the comparative evaluation of different representations for computer-based question-answering systems.

Three languages for representing such pictorial data are constructed. The basic units of the first are pictures, of the second trees, and of the third sentences. Each of the three languages is further modified to serve for describing data, for specifying constructions, for posing queries, and for stating answers. The interrelations among the various specialized uses of these three languages are investigated. Queries are best posed in an English-like language, computer search best proceeds on data represented as trees, and answers can often be best presented in picture representations. Results are in the form of a) context-free generative grammars for the different languages expressed as production rules, b) theorems showing correspondences between, say, all query sentences and all pictorial answers, and c) formula for the effort to search for answers, for optimal trees to store data.

TABLE OF CONTENTS

Section Page

I. INTRODUCTION 1

II. A PICTORIAL LANGUAGE 3

III. ANSWERING QUESTIONS AND EXECUTING CONSTRUCTION ORDERS GIVEN IN A PICTORIAL LANGUAGE. 12

IV. A TREE LANGUAGE. 17

A. The Tree-Language Specialized for Expressing Descriptions . 19
B. The Tree-Language Specialized for Expressing Constructions 21
C. The Tree-Language Specialized for Posing Queries 23
D. The Tree-Language Specialized for Stating Answers 26

V. OPTIMAL TREES FOR STORAGE AND SEARCH 30

VI. ENGLISH-LIKE LANGUAGES 34

A. An English-Like Language for Describing "Pictorial" Data . . 34
B. The English- Like Language for Specifying Constructions . . 39
C. The Name Language Specialized for Posing Queries and Stating Answers 45

VII. CONCLUDING COMMENTS 51

REFERENCES 52

I. INTRODUCTION

Since the potential of computers for non-arithmetic processes has been recognized, the problems of pictorial and linguistic data processing have attracted increasing interest. One way to increase the sophistication of the computer art in this direction is to present the computer with data in pictorial form and interrogate it in restricted English. To answer queries, the machine should be able to search its internal memory for data responsive to the query. This involves capability of processing at least 3 different languages: pictorial representation, representation as sentences in English, representation suitable for updating and searching the machine's memory.

A user may require the machine to construct, search, describe, or interrogate a given body of data; he may state his requirement in any one of the three languages; he may have fed the data into the machine in any one of them; and he may wish the response in any one of them. It is, therefore, of interest to formally study the relationship between these three languages. Can anything represented in one language also be represented in the other two? Can queries asked in one language be answered in another? What are the relative merits of these different languages for various purposes?

These questions are of some interest in themselves, though the answers are obvious for the highly restricted domain of discourse considered here. The techniques of answering them can, however, be extended as the domain of discourse is extended and as the languages are enriched. Both the techniques and the answers are useful in investigating the informational equivalence of two descriptions (e.g., if both lead to the same construction specifications), the relevance of answers to queries, in assessing the choice of different means of representation available to the designer of an information system.

Similar problems were studied by Kirsch[7], Simmons and Londe[8], Sutherland[9] to mention a few. The idea of using our restricted domain of discourse was suggested by S. Amarel (private communication) and mentioned by M. Minsky[10]. The use of trees for storage and search has been studied in more detail by S. Amarel[1] and R. McNaughton as an application of multi-computer systems[6].

We have not studied questions of translating from these languages into the predicate calculus, as being done, for example, by Bohnert[2], Cooper[4], Darlington[5]. Nor have we addressed ourselves to the important problem of how to get a machine to select significant conjectures, to pose deep questions, of condensing or summarizing information[3]. We will touch upon this problem in a forthcoming paper on methods for translating query sentences into computer search programs.

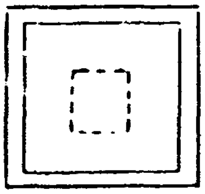
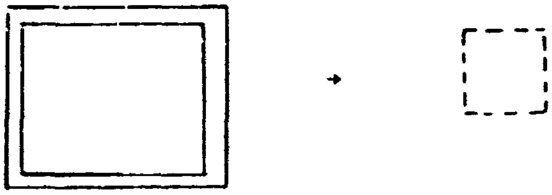
II. A PICTORIAL LANGUAGE

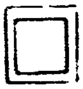
By a language L we mean the set of all possible sentences generated by a linguistic system S(L). A linguistic system consists of a quadruple

$$(V_T, V_N, U, R)$$

in which: V_T denotes the terminal vocabulary, which, for a pictorial language, consists of geometric objects such as ∩, ○, ▣, △ ; V_N denotes the non-terminal vocabulary, which, for a pictorial language, consists of configurations; U denotes a special element in V_N corresponding to the unit of the language, a complete graphic message or picture in a pictorial language; R denotes a set of production rules to be illustrated next for a pictorial language.*

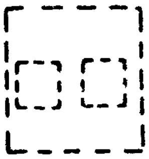
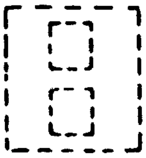
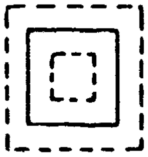
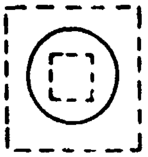
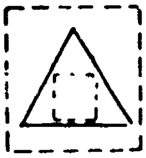
The first of these rules for the pictorial linguistic system asserts that U consists of:

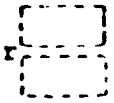
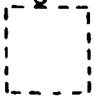
(1)  or: Rule 1: 

Rule 1 is understood as follows. A picture (corresponding to "sentence") consists of a double-line frame. Inside the frame is a "configuration", a rectangle with dotted lines. Unless otherwise indicated, the figures with dotted lines can be located anywhere inside the frame and have any size. Rule 1 states that one figure can be replaced by or produced from 

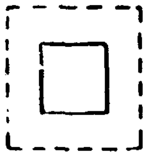
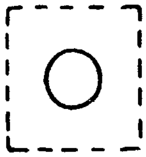
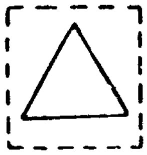
* The rules of replacement are formulated here in the reverse order to that conventionally used for phrase structure languages; i.e. the replacements here are "from specific to general" in contrast to replacements "from general to specific" that are common in phrase structure grammar formulations.

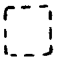
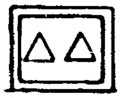
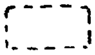
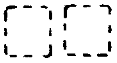
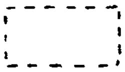
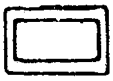
The second rule specifies how configurations may be formed.

(2a)  (2b)  (2c)  (2d)  (2e) 

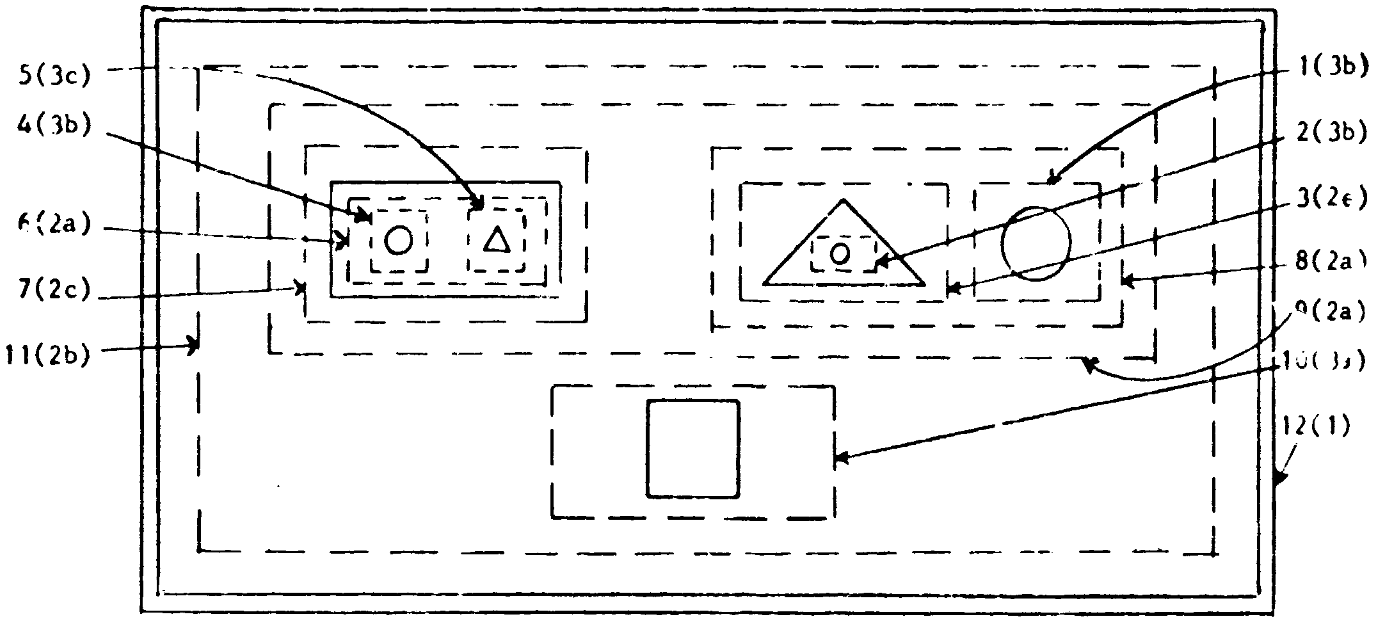
Unless otherwise indicated, the figures in solid lines can be located anywhere and have any size provided their edges do not intersect any other edges. Rule (2b), for example, states that any pair  can be replaced by  . Generally, the rules assert that whenever there is a figure having the form that remains when the outer dotted line is removed, that figure can be replaced by the corresponding outer square of dotted lines.

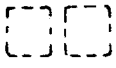
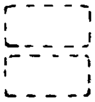
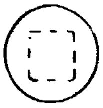
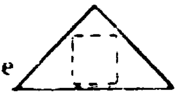
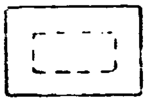
Rule 3 relates configurations to specific objects:

(3a)  (3b)  (3c) 

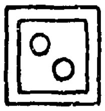
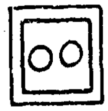
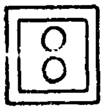
Rule (3c) asserts that a triangle of any size can be replaced by  . To verify that  is a picture, we apply Rule (3c) to the left triangle and form:  . We also apply (3c) again to the right triangle to get altogether:  . We now apply Rule (2a) to get . We now apply Rule 1 to get , and this terminates the verification that we have a picture. We can repeat this procedure without removing the figures being replaced, and numbering the order in which the rules were applied.

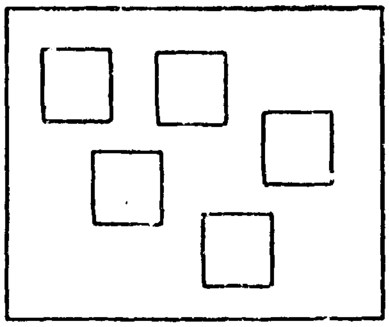
We take a more complex example.



The "terminal vocabulary" V_T consists of all squares, circles and equilateral triangles. The "non-terminal vocabulary" V_N consists of all rectangles made of dotted lines, all figures like , like* , like , like , like  and of all rectangles with double edges. The latter plays the roles of a picture-designator U. The rules R all consist of a figure of V_N enclosing a figure of either V_N or V_T, stating that the enclosed figure can be replaced by the enclosing figure.

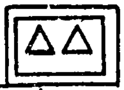
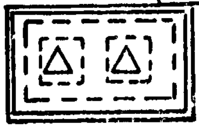
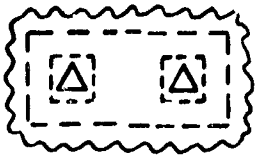
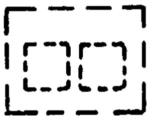
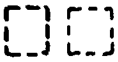
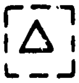
* This corresponds to the relation expressed by "below", used at the end of this section.

The accompanying figure is not a picture according to the above rules. Note also that the picture  could not be distinguished from either  or  . Rules 1-3 specify a particular pictorial description language L_{GD}, which excludes many figures we would realistically like to call "pictures". By augmenting V_T, V_N, and R of the system $S(L_{GD})$, we can enable L_{GD} to better approximate reality. But this is not our primary intent here. In the language L_{GD} specified by the above V_T, V_N, R there is no limit to the number of geometric objects which may be in one picture.

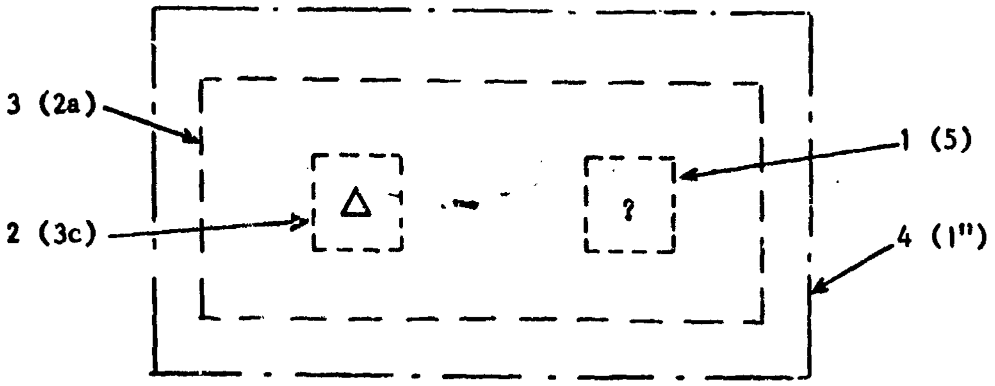


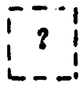
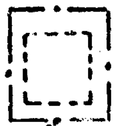
By a construction language we mean the set of all possible imperative "sentences" generated by a linguistic system. Each imperative "sentence" is a command specifying a construction. In the case of a pictorial construction language L_{GC}, each "sentence" is in a form corresponding to a "parsed" picture. The rules R_{GC} are the same as R_{GD}, except that they require the interior of a dotted-line square to replace the outer boundary rather than vice versa. Only Rule 1 differs in that it uses a wavy line instead of a double line boundary.

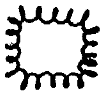
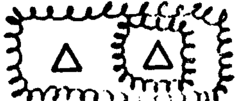
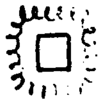
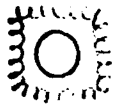
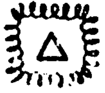
Each rule of R_{GC} specifies a construction step. It consists of erasing dotted lines in a surrounding rectangle, or replacing the wavy line by a double line frame, or terminating when the objects of V_T are reached. Both V_T and V_N are the same for L_{GC} as they were for L_{GD}.

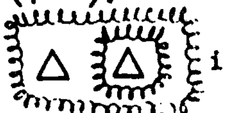
As an example, suppose we wish to represent an order to construct (copy) the picture . The order would first be described in L_{GD}, as follows: . The following figure  is the corresponding "imperative sentence" of L_{GC}. To check that this is a well-formed member of L_{GC} we proceed by: 1) applying the rule  to get ; 2) applying the rule  to the pair to get △△, 3) replacing the wavy line of the frame by the double line. It is important to keep in mind the distinction between rules legitimizing the form of construction orders and rules for executing such orders.

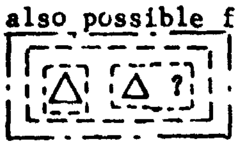
By a query language L_Q we mean the set of all possible interrogative "sentences" generated by a system $S(L_Q) = (V_{NQ}, V_{TQ}, U_Q, R_Q)$. Each query "sentence" states what is wanted and what is known. It always makes implicit reference to a corpus to be searched. The corpus is a subset of L_{GD} not L_{GD} itself. In a <u>graphic</u> query language, the queries are again in a form corresponding to pictures. Instead of the double-line frame which denoted a picture to be analyzed in L_{GD} or a wavy-line frame which denoted a picture to be copied in L_{GD} we use a dot-dash-line frame to denote a pictorial query. Question-marks are in the place of objects in V_T. Answers consist of pictures in L_{GD} with all question-marks replaced by objects in V_T. The linguistic system for L_{GQ} is the same as that for L_{GD} except that: in Rule 1, the double-line frame is replaced by a line like ——·——·—— ; the object ? is added to V_T. The rules are unchanged. The following figure shows the order of the steps and the rules needed to verify that it is a pictorial query.

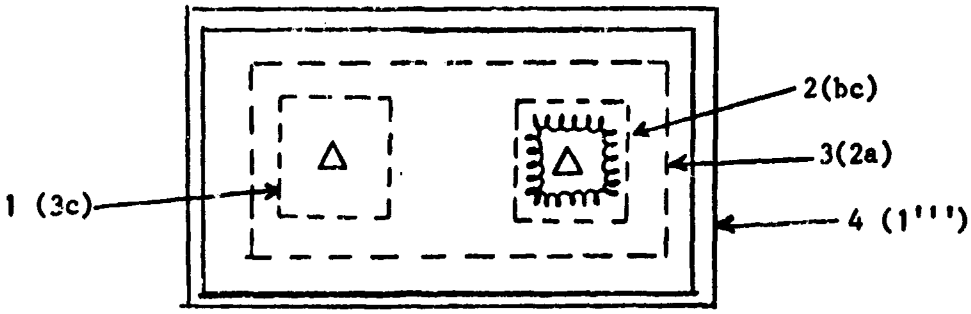


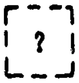
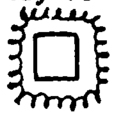
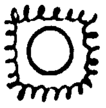
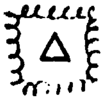
Here* (5) is:  and (1") is  , which is U.

By a <u>processing</u> or <u>search</u> or <u>answer</u> language L_A, we mean the set of all answer-sentences generated by a system $S(L_A)$. A <u>pictorial</u> answer sentence is a picture of L_{GD} in which the object replacing ? in a pictorial query of L_{GQ} is surrounded by a frame like . The outer frame is similarly replaced. Thus,  is an answer-sentence. The system $S(L_{GA})$ is the same as that for $S(L_{GD})$, except that  ,  and  are added to V_T and 3 corresponding rules are added to rule 3, and rule 1 is replaced by (1''').

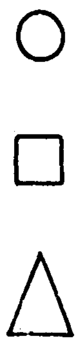
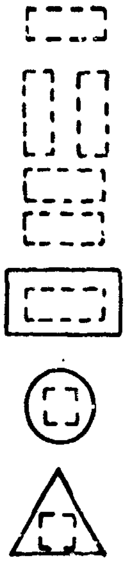
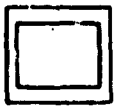
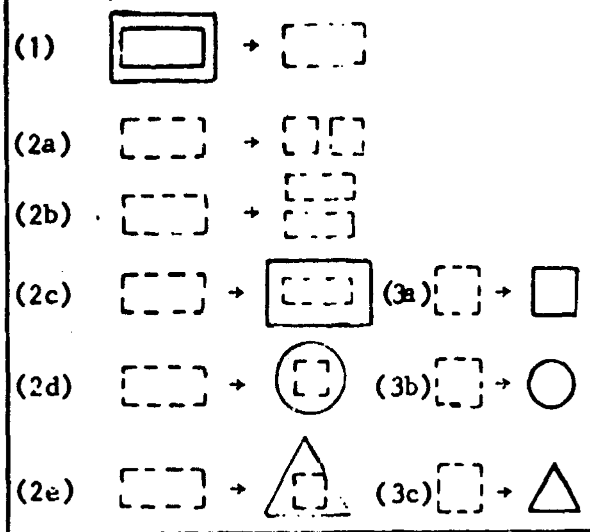
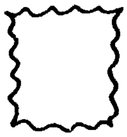
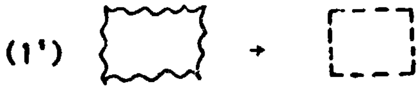
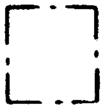
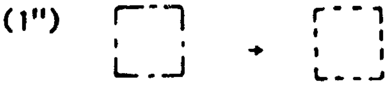
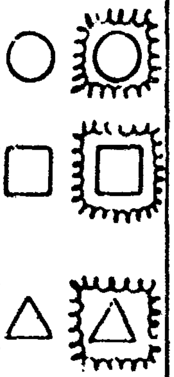
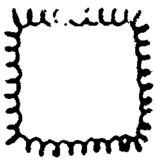
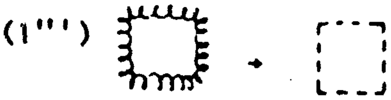
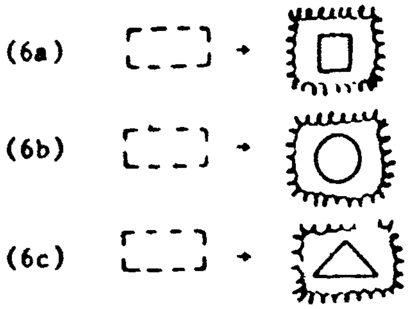
To verify that  is an answer-sentence of L_{GA}, proceed in the order shown in the accompanying figure.

* It is also possible for both an object (e.g.,△) and ? to appear in a box, e.g.,  . This is also a pictorial query. It asks for verification of the figure. Both rules (3c) and (5) apply to the right inner box. Except for the combination of ? with objects, only one rule can apply to a dotted-line box.

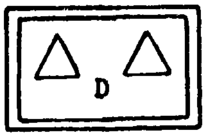


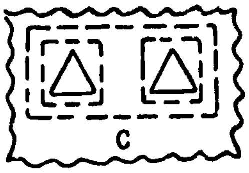
To answer a pictorial query is to produce an answer-sentence in which  is replaced by  or  or  by first producing a construction sentence.

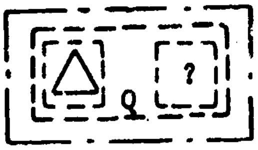
We summarize the four aspects of our pictorial language system below:

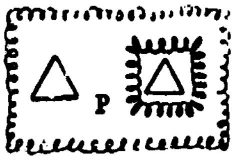
	Terminal Vocabulary V_T	Non-terminal Vocabulary V_N	Unit of the Language U	Rules of Formation R (for verifying membership in L)
$S(L_{GD})$ Pictorial Description Language				 (1) (2a) (2b) (2c) (3a) (2d) (3b) (2e) (3c)
$S(L_{GC})$ Pictorial Construction Language	same as above	same as above		 2a-2e, 3a-3c as above plus (4) check that all the dotted lines produced by substitution are already there.
$S(L_{GQ})$ Pictorial Query Language	same as above Plus ?	same as above		 2a-2e, 3a-3c, 4 as above plus (5) ?
$S(L_{GA})$ Pictorial Answer Language		as above, not ?		 2a-2e, 3a-3c as above plus 

Consider the following 4 examples:









To verify that:

$D \in L_{GD}$, apply, in order, rules (3c), (3c), (2a), (1)

$C \in L_{GC}$, apply, in order, rules (4), (3c), (3c), (2a), (1'),

$Q \in L_{GQ}$, apply, in order, rules (4), (3c), (5), (2a), (1")

$P \in L_{GA}$, apply, in order, rules (3c), (6c), (2a), (1"')

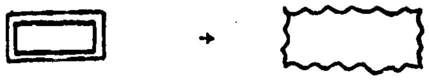
III. ANSWERING QUESTIONS AND EXECUTING CONSTRUCTION ORDERS GIVEN IN A PICTORIAL LANGUAGE

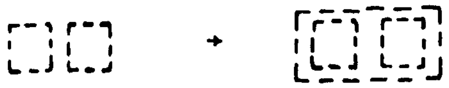
We now provide a totally different set of rules for:

Applicable Procedure — **Rules**

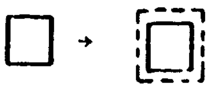
Executing Construction:

e.g., Given C, apply C1, C2a, C3c, C3c, to produce D, the unit of L_{GD}

C1. 

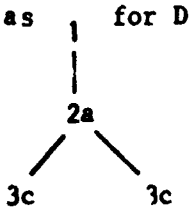
C2a. 

similarly for C2b-C2e; reverse arrow of 2a-2e.

C3a.  and similarly for C3b, C3c and C6a, C6b, C6c.

Checking that result of construction is as specified.

By a parsing tree we mean a tree such as



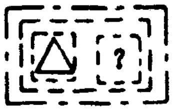
for D above.

(Ch 1) Parse result using (1)-(3c), i.e., record the rules and order in which they are used, deleting rule (1).

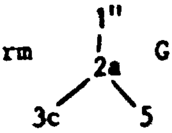
(Ch2) Delete rules (1') and (4) from parsing of construction statement (e.g.,C).

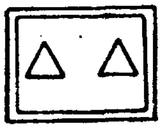
(Ch3) Check that parsing of result = parsing of construction statement to within partial order.

Answering a query.

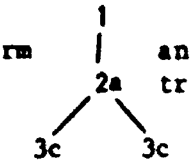
Example: Given: 

Form



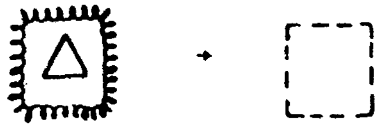
Given: 

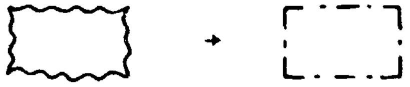
Form



and search all such trees.

This tree matches the above except that 3c is in place of 5. Form C6c which is



(Q1) 

(Q2) Form parsing tree for query; delete rule (1").

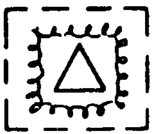
(Q3) Search all parsing trees in a corpus for pictures with (1) deleted, until one is found which matches that of query except for rule (5).

(Q4) Suppose rule 3x, x=a. b, c holds in place of rule (5). Form rule C6x.

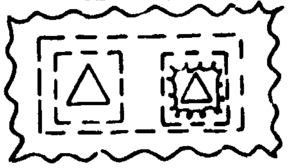
(Q5) Put C in front of all rules used in query, with C6x replacing rule (5).

Apply Rules

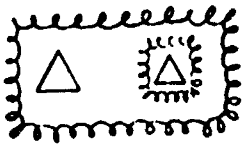
C3a, C3c, C6c e.g. applying C6c yields



Form

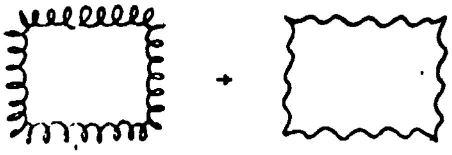


Form



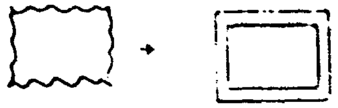
(Q6) Form a construction statement from all the rules used in the query by combining the dotted lines on the left side of a rule with what they replace.

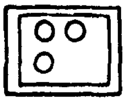
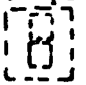
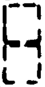
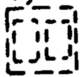
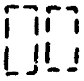
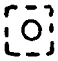
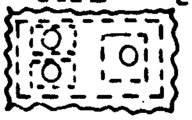
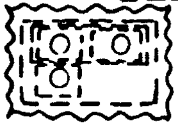
(Q7) Execute the above construction rules.

(Q8) 

Note that all the rules of formation for the pictorial language we have written are in the form of production rules. We have generalized from the conventional notion of concatenation used in linguistics -- which means placing two one-dimensional strings next to each other -- to where it can also mean adjoining two-dimensional arrays into horizontal, vertical, or enclosure adjacency relations. This casts our pictorial language clearly into the class of context-free languages, with our extended interpretation of concatenation. All the notions and results, including the problems of structural ambiguity, apply to our language. The following results all derived from this.

Theorem 3.1: To every picture in L_{GD} corresponds at least one pictorial construction statement in L_{GC}.

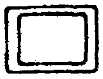
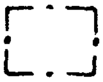
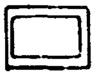
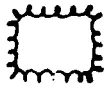
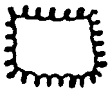
Proof: Given a figure like D, identify the elements in V_T in it. Surround each by a square of dotted lines. Then apply the rules (Q6) which were used in forming a constructing statement plus the rule  .

It is easily verified that the result is an element of L_{GC}. Because there is, in general, a choice in which rules (Q6) can be applied, more than one construction statement corresponds to a given picture. For example, if the picture is  , and we have rules: (C'2b):  →  , (C'2a)  →  ; and (C'3a):  → ○ , we get both  and  depending on whether we apply the rules (C'3a) 3 times, (C'2b), (C'2a) or (C'3a) 3 times, (C'2a), (C'2b). If the given figure is not an element of G_{GD}, rules (C'2a) and (C'2b) will not apply because rules 2a and 2b do not apply.

Theorem 3.2: To every pictorial construction statement in L_{GC} corresponds a unique picture in L_{GC}.

Proof: Given any element of L_{GC}, supply rules C1, C2a, C2e, C3b, C3b, C3c, where applicable in that order. Apply Rules (Ch 1) (Ch 2) (Ch 3) to verify that the result of the construction is as specified. Since applying Rule Ch 1 involves parsing the result by the rules of $S(L_{GD})$, this verifies that the result of construction is in L_{GD}. The order of applying rules C1-C3c is specified. (Applying these rules is tantamount to erasing the dotted lines, from the outside in). Hence, the resulting picture is unique.

Theorem 3.3: To every picture D in L_{GD} which contains n objects of V_T correspond at least 2^n queries in L_{GQ} each having answer in L_{GA} corresponding to D.

Proof: Given a picture K, we form a query by: 1) parsing D according to the rules of $S(L_{GD})$ leaving in all dotted lines with what replaces them; 2) replacing  by  3) replacing either one, two, three, or all n, of the objects in D by ?. Step 3 can be done in $\binom{n}{1} + \binom{n}{2} + \ldots + \binom{n}{n} = 2^n$ ways. For each way, there is at least one parsing of D. To verify that the result of this construction is in L_{GQ}, apply the rules of $S(L_{GQ})$. Applying steps (Q1)-(Q8) results in an element of L_{GA}. This is verified by applying the rules of $S(L_{GA})$. To check that this is an answer, replace the outer frame according to  →  and delete  on the inside wherever it occurs. The result is identical with D.

Theorem 3.4: Consider any query Q in L_{GQ} with an associated corpus Corp (Q), which is a finite subset of L_{GD}, containing m pictures. Suppose that a fraction f of the m pictures correspond to answers for Q, each of the m pictures having the same probability of corresponding to an answer. If $f > 0$, it will take, on the average, $f/2[m^2 + 3m - f(2m^2 + 3m) + f^2m^2 + 2]$ search comparisons to find an answer. If $f = 0$, it will take m comparisons to ascertain that Q has the answer: "Query specifications are not met."

Proof: To answer Q, execute steps (Q1)-(Q8) to produce a pictorial answer. To check that this is in L_{GA} and is an answer proceed as in the proof of Th. 3. Because the corpus to be searched in step (Q3) is finite, the procedure will terminate in a finite number of steps. If, in step (Q3), the parsing tree of Q does not match any tree in the corpus, all m such trees will have been checked to ascertain that the query specifications are not met. If there are fm pictures in the corpus which correspond to an answer, then the probability that the i^{th} picture in some ordering of the m pictures of the corpus is an answer is $\frac{fm}{m}$. If we examine all the pictures of the corpus in order, this is the probability of stopping at the i^{th}, and the expected number of pictures examined before the first match is $\Sigma\, i \cdot f$. The sum goes from $i=1$ to $i=m-fm + 1$, because in the extreme case all the fm answers are in a row at the end. The sum is $f/2[(m-fm + 1)(m-fm +2)]$, which is $f/2[m^2+3m-f(2m^2+3m) + f^2m^2 + 2]$.

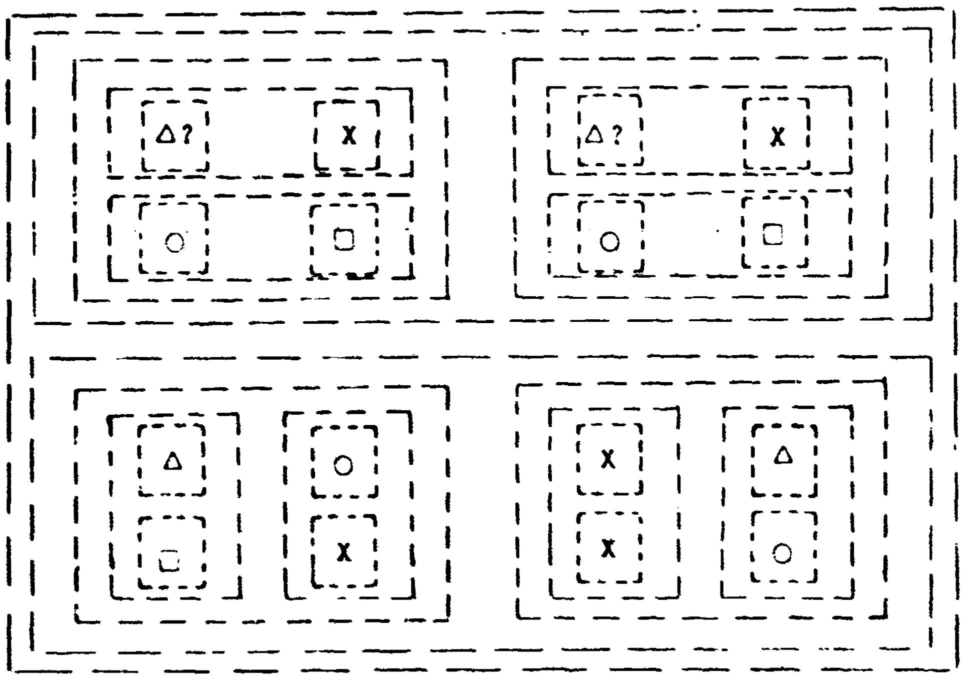
If Q has a single question-mark, the corpus could contain at most 3 answers, corresponding to □ , ○, or △ in place of ?. If Q has k question marks, the corpus could contain at most 3^k answers. Let g(m) be the fraction of "possible" answers which are in the corpus. Thus, fm = $g(m)\cdot 3^k$. If, for example, $g(m) = \frac{m}{N+m}$, then the expected number of searches increased with m and k approximately as

$$\frac{m^2 + 3m}{2(N+m)} \cdot 3^k - \frac{2m^2 + 3m}{2(N+m)^2} \cdot g^k$$

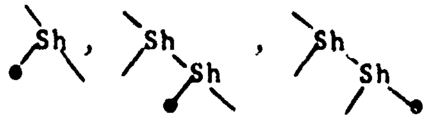
If Corp (Q) is not finite, the answer-procedure may not terminate in a finite number of steps, depending on the decidability of Q.

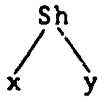
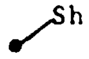
We conclude by illustrating a pictorial query which corresponds to the question: "In a given set of pictures which are in a given corpus, is it true that each circle which is to the left of a square is below a triangle?"

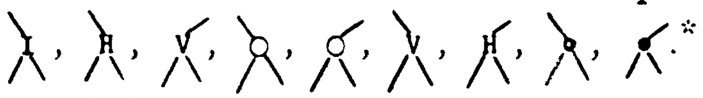
Boxes marked by X can have any of the three objects of V_T inside.

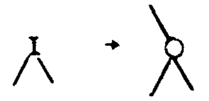


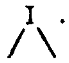
IV. A TREE LANGUAGE

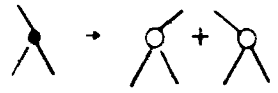
The terminal vocabulary of the tree-language system which we will carry as our running illustration, contains: 

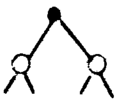
(Sh is mnemonic for "Shape"). These labeled trees correspond to ○ , □ , and △ in the terminal vocabulary of the pictorial language system. We use them so that the tree-language consists exclusively of trees, just as the pictorial language consisted exclusively of diagrams. It would be just as well to use ○, □ , △ in place of these trees, as terminal nodes. To motivate the use of these particular trees, we read  as stating "each polygon in configuration x has more corners than any polygon in configuration y." A dot, as in  denotes a specific object.

The non-terminal vocabulary of $S(L_T)$ contains labeled trees like * We state next some of the rules for $S(L_T)$.

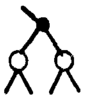
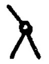
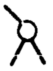
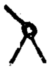
Rule T2: 

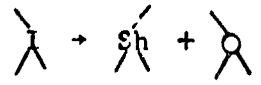
This rule states that any tree with H, I, V or Sh in place of the circle can be replaced by  .

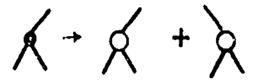
Rule T3a: 

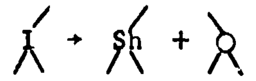
The right-hand side of this rule designates a pair of trees both joined at the same node above them, as  . As before, ○ stands for H, I, V or Sh. The left-hand tree is any tree with H, V but not Sh or I, in

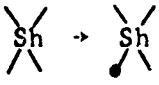
* To facilitate reading, we remind the reader that I indicates "enclosure", H "to the left/right of", V "above/below", Sh "Shape", ○ a variable for H, I, V or Sh; ● a variable for just H or V.

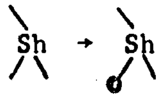
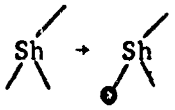
place of the dark circle. The rule states that the tree  can be replaced by . Rule T2 also applies to this tree for whenever a rule holds with  on the right-hand side, it also holds for .

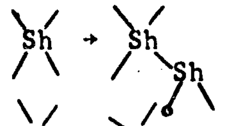
T3b) 

T3c) 

T3d) 

T4a) 

T4a actually stands for two rules,  and 

T4b) 

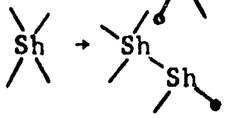
T4c) 

Figure 4.1 is a tree formed according to these rules, Fig. 4.2 is not. To verify that Fig. 4.1 is in L_T apply rule (T4a) four times to the bottom of Fig. 4.1. Then, substituting white circles for the Sh and black circles for the V, apply rule T3a to the right side of Fig. 4.1; substituting white circles for the Sh and the black circles for I, apply rule (T3d) to the left side of Fig. 4.1; with I and V in place of the two white circles

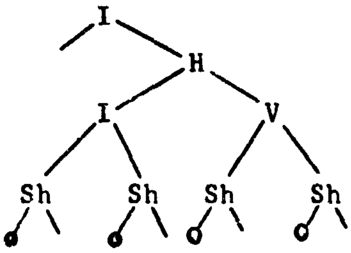


Fig. 4.1

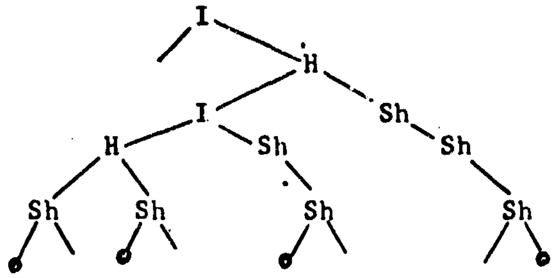


Fig. 4.2

on the right-hand side if rule T3a and H in place of the black circle on its left, apply the rule to get $\overset{\diagdown}{\underset{/\backslash}{H}}$. With H in place of the white circle, apply rule T2.

As in the case of the pictorial language, the rules of formation for the tree language, with a suitable extension of the idea of concatenation, cast the grammar of the tree-language in the context-free class. The questions of recognizing well-formed formulas in the language are thus special cases of context-free languages in general. It is, however, of interest to examine the tests for well-formedness in more detail, for more specialized versions of the tree-language. We should also like to look at the connections between these specialized versions and their correspondents in the pictorial language.

A. The Tree-Language Specialized for Expressing Descriptions

We now add a distinguished element to the non-terminal vocabulary of $S(T_{LD})$. We call it D. We now introduce rule TD1: $D \rightarrow \underset{/\backslash}{I}$. Actually, D will appear as ${}_{D}\overset{I}{/\backslash}$, and when rule TD1 is applied, $\underset{/\backslash}{I}$ is erased and D remains. This marks completion of the verification that the tree is a member of L_{TD}. We will also call such trees D-trees. By a parsing of a D-tree we will mean a diagram showing the rules used, and the order of their use. If we complete Fig. 4.1 by adjoining D to the upper left part of the tree, the use of rules to verify that Fig. 4.1 is in L_{TD} can be described in Fig. 4.3. Rules appearing on the same line in this tree are applied simultaneously; the order is immaterial; rules on upper levels are applied after rules on lower levels. We will call this a parsing tree.

Theorem 4.1: To each element of L_{GD} corresponds at least one element of L_{TD}.

Proof: We will show that the rules of S(L_{GD}) and S(L_{TD}) are equivalent under appropriate identifications. Identify $\overset{\text{Sh}}{\bullet\!/\backslash}$ with ○, $\overset{\text{Sh}}{/\,\searrow}\overset{}{\text{Sh}}$ with □ and $\overset{\text{Sh}}{/\,\searrow}\text{Sh}$ with △. Next, identify D with ▭. Next, identify ⬚ with $\overset{\text{I}}{/\backslash}$, $\overset{\circ}{/\backslash}$ and $\overset{\circ}{/\backslash}$, where $\overset{\text{I}}{/\backslash}$ is such that only D or $\overset{\text{Sh}}{/\backslash}$ can hang on its left bottom branch. Identify ⬚⬚ with $\overset{\text{H}}{/\backslash}$, and $\overset{\text{H}}{/\backslash}$, also ⬚ with $\overset{\text{V}}{/\backslash}$ and $\overset{\text{V}}{/\backslash}$ and □, ○, △ all with $\overset{\text{I}}{/\backslash}$, where $\overset{\text{I}}{/\backslash}$ is again such that only $\overset{\text{Sh}}{/\backslash}$ can hang on its left bottom branch.

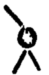
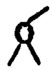
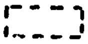
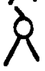
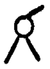
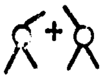
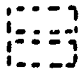
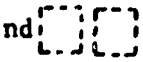
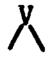
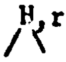
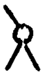
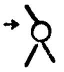
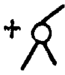
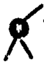
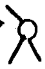
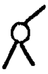
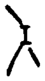
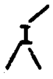
Rule (1) of L_{GD} which states ▭ → ⬚ thus corresponds to D → $\overset{\text{I}}{/\backslash}$ or rule TD1. Rule (2a) of L_{GD} stated: ⬚ → ⬚⬚. This now becomes: (i) $\overset{\text{I}}{/\backslash}$ → $\overset{\text{H}}{/\backslash}$, using the above identifications of ⬚ with $\overset{\text{I}}{/\backslash}$. There is nothing like $\overset{\text{I}}{/\backslash}$ → $\overset{\text{H}}{/\backslash}$ because we allowed only $\overset{\text{Sh}}{/\backslash}$ to hang in the left bottom branch of $\overset{\text{I}}{/\backslash}$. Rule (2b) of L_{GD} stated: ⬚ → ⬚. This now becomes:

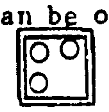
(ii) $\overset{\text{I}}{/\backslash}$ → $\overset{\text{V}}{/\backslash}$. Rules (2c) - (2e) of L_{GD} all become:

(iii) $\overset{\text{I}}{/\backslash}$ → $\overset{\text{I}}{/\backslash}$.

We now use the identification of ⬚ with $\overset{\circ}{/\backslash}$, specializing the latter to $\overset{\text{Sh}}{/\backslash}$. Rule (3a) of L_{GD} which is ⬚ → □, becomes $\overset{\text{Sh}}{/\backslash}$ → $\overset{\text{Sh}}{\bullet\!/\backslash}$. This is half of rule T4a. The other half, $\overset{\text{Sh}}{/\backslash}$ → $\overset{\text{Sh}}{\bullet\!/\backslash}$ is obtained by identifying ⬚ with $\overset{\circ}{/\backslash}$. In a similar manner, we can identify rules (3b) and (3c) of L_{GD} with T4b and T4c.

We now identify ⬚ with $\overset{\text{I}}{/\backslash}$ once more and apply rules (3a), (3b), (3c) of L_{GD} again. We wish to show that $\overset{\text{I}}{/\backslash}$ → $\overset{\text{Sh}}{/\backslash}$ follows. But (3a), (3b), (3c) of L_{GD} would result in $\overset{\text{I}}{/\backslash}$ → $\overset{\text{Sh}}{/\backslash}$, $\overset{\text{I}}{/\backslash}$ → $\overset{\text{Sh}}{/\,\searrow}\text{Sh}$, $\overset{\text{I}}{/\backslash}$ → $\overset{\text{Sh}}{/\,\searrow}\text{Sh}$ just as if Sh were substituted for I. We write this substitutability condition as (iv) $\overset{\text{I}}{/\backslash}$ → $\overset{\text{Sh}}{/\backslash}$. Now, (i), (ii), (iii), (iv), can all be put together to state $\overset{\text{I}}{/\backslash}$ → $\overset{\circ}{/\backslash}$, which is rule T2.

It remains only to verify that rules T3(a-d) follow from the rules of L_{GD}. The symbols  and  are each identified with , since they are special cases of , . With  we identify both  and , for which we can substitute  and  respectively. Hence  →  +  and  →  +  follow. This leaves only (T3b) and (T3d) to derive, and this follows directly from the definition of  and , namely, that only Sh can be attached to the left bottom branch. Hence all the rules of $S(L_{ID})$ follow from the rules of $S(L_{GD})$. No rules are implied by $S(L_{GD})$ which are not in $S(L_{TD})$.

A given picture in L_{GD} can be obtained from the rules of $S(L_{GD})$ in at least one way. For example,  can be obtained by applying rules (3b),(3b),(3b),(2b),(2a) and (1) or by applying rules (3b)(3b)(3b)(2a) (2b) and (1). This corresponds to the two D-trees: Figs. 4.4 and 4.5.

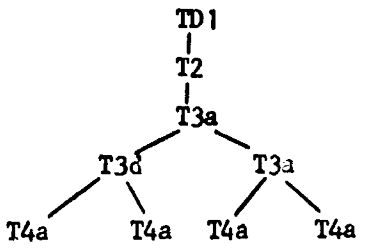


Fig. 4.3

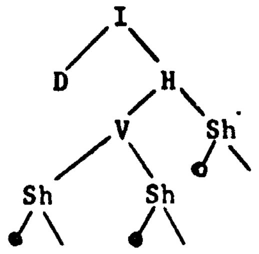


Fig. 4.4

and

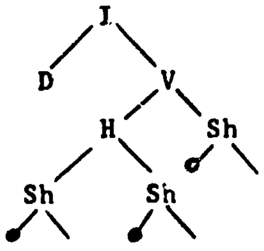


Fig. 4.5

This proves the theorem.

B. The Tree-Language Specialized for Expressing Constructions

The distinguished element of the non-terminal vocabulary of $S(L_{TC})$ which corresponds to the unit of L_{TC} is designated by C. Otherwise, the terminal and non-terminal vocabulary is as it was for $S(L_{TD})$. All rules are unchanged except TD1 which goes to TC1: C → . A construction-tree thus looks exactly like a D-tree, except that D is replaced by C.

We must add one more rule, however. In verifying that a given tree belonged to L_{TD}, we replaced the tree on the right-hand side of a rule by the left-hand side of the rule, and completed verification when the process ended with symbol D. Now we will not replace the right-hand side of a rule by the left-hand side, but simply check that the right-hand tree indicated on the side of the rule is properly attached to the tree on the left-hand side of the rule, we call this rule TC5.

Theorem 4.2: The construction language L_{GC} and L_{TC} are in 1-1 correspondence.

Proof: Recall that an element of L_{GC} is a picture with every configuration enclosed in a rectangle of dotted lines, and a frame of wavy lines. It is easy to see that TC1 corresponds exactly to rule (1) of $S(L_{GC})$ and TC5 exactly to rule (4) of $S(L_{GC})$. The other rules of $S(L_{GC})$ are the same as those for $S(L_{GD})$ and those of $S(L_{TD})$ and $S(L_{GD})$ have been established in the preceding theorem.

It remains to show that to each tree in L_{TC} corresponds a unique element of L_{GC}. The tree in L_{TC} specifies a particular sequence of rules to be applied in a specified order, except for rules at the same level. These rules of $S(L_{TC})$ correspond uniquely to rules of $S(L_{GC})$, to be applied in the same order. This generates exactly one "parsed" diagram or element of L_{GC}.

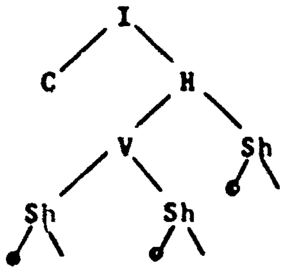


Fig. 4.6

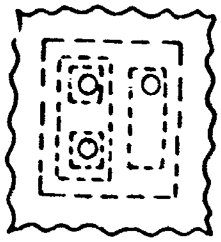


Fig. 4.7

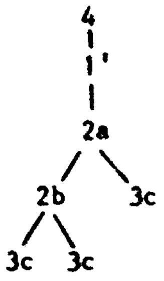


Fig. 4.8

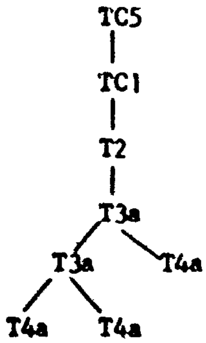


Fig. 4.9

For example, the tree in Fig. 4.6 results in the construction specification shown in Fig.4.7. The parsing trees of rules are shown in Figures 4.8 and 4.9.

Theorem 4.3: To any construction specification tree in L_{TC} corresponds a unique picture in L_{GD}.

Proof: This follows from theorem 4.2 and theorem 3.2.

We first construct a pictorial construction statement and then proceed to construct the picture with rules C1-C3c of Section III.

C. The Tree-Language Specialized by Posing Queries

In parallel to the study of previous subsystems we add to V_N the distinguished element designed by Q, and to V_T the element ?. The rules of $S(L_{TQ})$ are the same as those of $S(L_{TC})$, except that rule TC1 is replaced by: Rule TQ1: $Q \rightarrow \underset{/\backslash}{I}$. A typical query-tree is shown in Fig. 4.10.

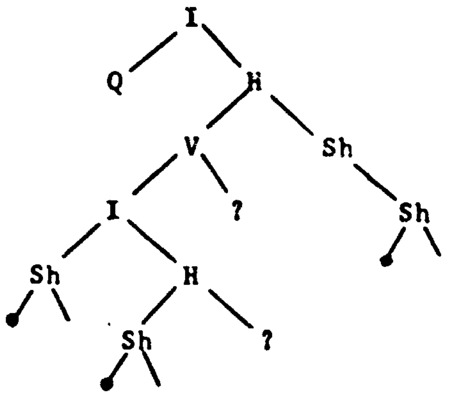


Fig. 4.10.

We need only identify Q with ⬚ to be able to state the following theorem:

Theorem 4.4: To each query-tree of L_{TQ} corresponds a unique pictorial query in L_{GQ}.

The Q-tree shown in Fig. 4.10, for example corresponds to the pictorial query in Fig. 4.11.

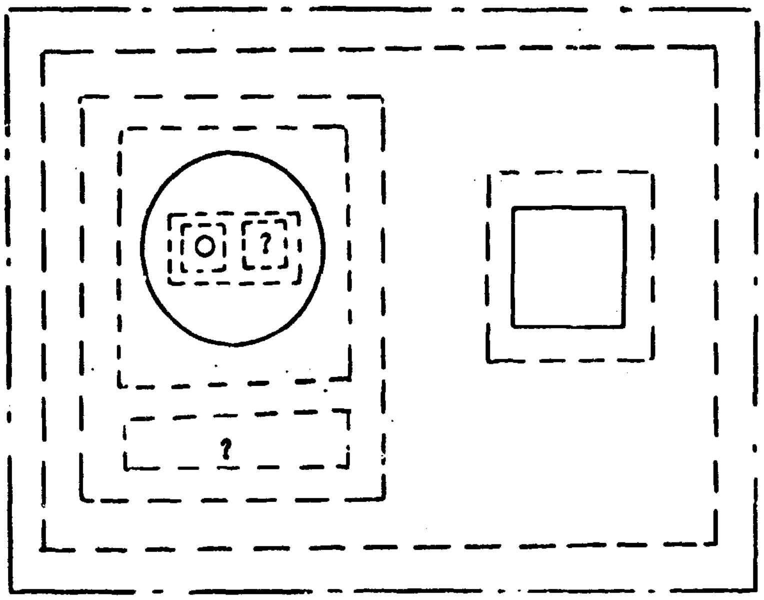


Fig. 4.11

We now wish to extend L_{TQ} in order to take advantage of the fact that a tree embodies many implications due to the ordering of certain relationships. We will introduce rules that allow us to form a query in which we replace a path in a D-tree which has $\overset{H}{\wedge}$ as vertex by the path $\overset{H'}{\wedge}$ connecting the same end-points. Similarly we introduce a corresponding rule for $\overset{V'}{\wedge}$ and $\overset{I'}{\wedge}$ Call these rules TQ6a, TQ6b and TQ6c. Thus we can replace

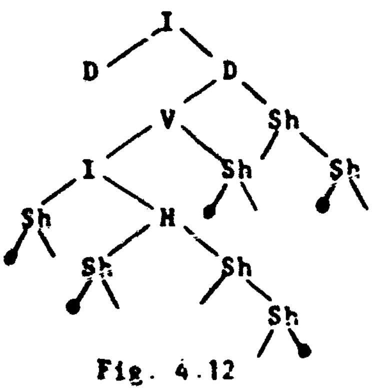


Fig. 4.12

by

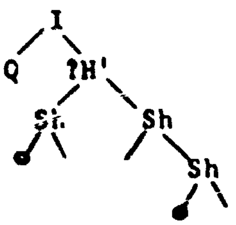


Fig. 4.13

or by

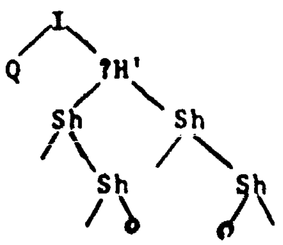


Fig. 4.14

or by

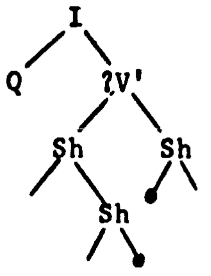


Fig. 4.15.

or by

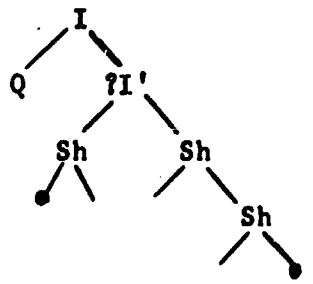


Fig. 4.16.

In words, these would ask: Is there a circle to the left of a square? Is there a triangle to the left of a square? Is there a triangle above a circle? Is there a triangle inside a circle? If a Q-graph has question-marks along side a node name, the relationship indicated is to be verified rather than filled in.

We shall call the augmented tree-query-language L^1_{TQ}.

Unlike the answering-procedure used within a graphic language, we do not search the corpus of parsing trees but the corpus of D-trees themselves. A search procedure somewhat analogous to Q1-Q8 in **Section III follows.**

TQS1: Apply rules TQ6a, TQyb, TQ6c where applicable to all D-trees in the corpus.

TQS2: Compare the transformed query tree with each D-tree in the given corpus ignoring: 1) a failure to match between nodes where there was a ? in the Q-tree 2) a failure to match Q and D. Thus, where Q-tree of Fig. 4.10 matches the D-tree of Fig. 4.12. Rule TQS1 did not apply.

If the Q-tree were that of Fig. 4.13 instead of Fig. 4.10, rule TQS1 would have been applied, and Fig. 4.12 would have been transformed into Fig. 4.17, among the many transformations that would have been possible. This transformation results in a match.

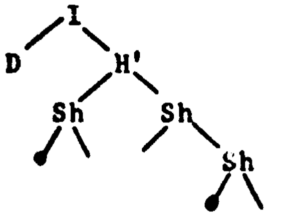


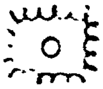
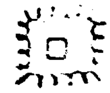
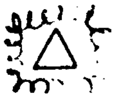
Fig. 4.17

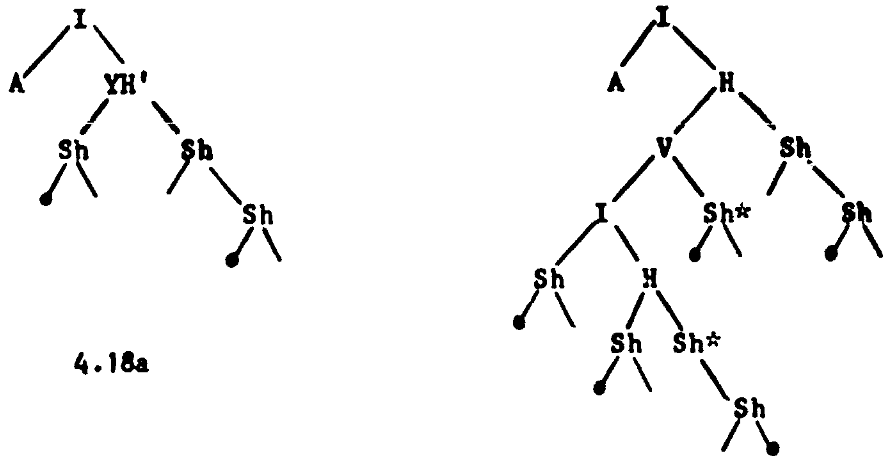
TQS3: Apply the rule C → Q, i.e., replace Q by C in all Q-tree. Also replace each question mark in the Q-tree by either: 1) the subtree of the matching D-tree which makes the match complete (except for D and C); this subtree will begin with $\overset{Sh}{/\backslash}$ and it will be marked $\overset{SH*}{/\backslash}$ 2) the symbol Y (to indicate "Yes") is the question-mark was next to a node as in Fig. 4.13, and the node label was the same in the Q-tree as in the matching D-tree; 3) the symbol N, (to indicate "No") if in the above case, the node label next to ? in the Q-tree is different from that in the matching D-tree. 4) The symbol M if there is no matching D-tree; also replace the vertex label by the one in the matching but non-verifying D-tree.

TQS4: Applying the above rule results in a C-tree, specifying a construction. Execute the indicated constructions. This results in an answer-tree.

D. The Tree-Language Specialized for Stating Answers

Two answer-trees are illustrated in Fig. 4.18. The procedure for constructing an answer-tree from a C-tree is to simply replace C by A and to copy the rest of the C-tree.

By identifying $\overset{Sh*}{/\backslash}$, $\overset{Sh*}{/\ \ \overset{Sh}{/\backslash}}$, $\overset{Sh*}{/\ \ \overset{Sh}{/\backslash}}$ with , ,  respectively we can obtain a 1-1 correspondence between answer-trees of L_{TA} and the pictorial answers of L_{GP} we understand the set of all possible



4.18a

4.18b

Figure 4.18

trees of the type illustrated in Fig. 4.18b. There is no pictorial answer corresponding to Fig. 4.18a. The sublanguage L^1_{TA} introduced trees like that of Fig. 4.18a and 4.18b. In the sense that L^1_{TQ} and L^1_{TA} contain trees not in L_{TQ} and L_{TA} respectively, these are more powerful languages.

We could try to extend the graphic languages L_{GQ}, L_{GA} so that they correspond more closely to L^1_{TQ} and L^1_{TA}, as illustrated in Figs. 4.19 and 4.20. The pictorial query corresponding to Fig. 4.13 would be:

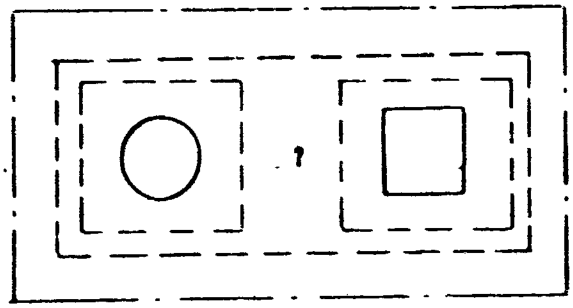


Fig. 4.19

The question-mark appearing in the outer box of dotted line indicates that verification of the relationship indicated by that box is in question.

The pictorial answer corresponding to Fig 4.21 is shown in Fig. 4.20. The curly frame around the vertical configuration consisting of two circles

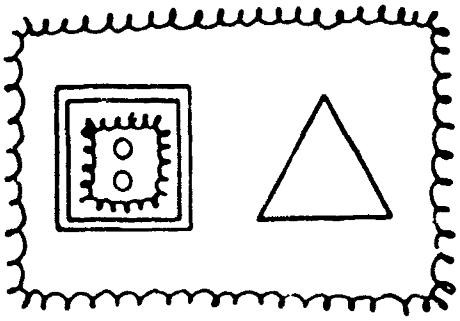


Fig. 4.20

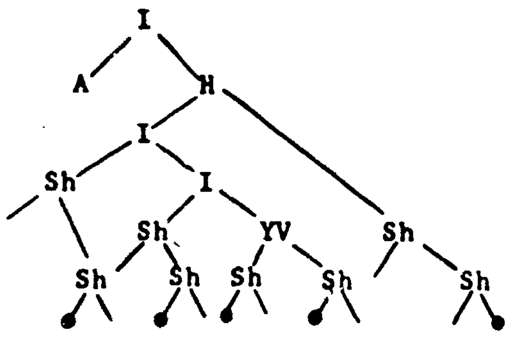


Fig. 4.21

indicates the answer to the questioned relation. If in Fig. 4.21, there were NV (to indicate "no, it is vertical") in place of YV (to indicate "yes, it is vertical"), then Fig. 4.20 could still be an answer-tree, but to a query which had, say, ? H instead of ? V at the corresponding node. But there is no way of representing the relations H^1, V^1 and I^1 in the pictorial languages. From this point of view, the tree-languages have greater power of representing queries and answers than do the graphic languages.

Theorem 4.5: There exist query-trees and answer-trees in a tree language for which there are no corresponding representations in the pictorial languages, L_{GS} and L_{GA}.

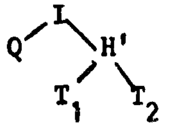
Step TSQ2 of the preceding section is critical for taking advantage of a tree-language. In the first place, it is well to note that in query-answering even in the pictorial languages, we compared trees -- the parsing trees made up of rules corresponding to pictures of L_{GD}.

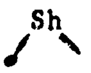
In step TSQ2, we ignore the Q and D as well as question-marks in comparing a given Q-tree with a corpus of D-trees. Hence we will strip the trees of the corpus and of the query of its question-marks and of $_{Q}\overset{I}{\diagup\diagdown}$ and $_{D}\overset{I}{\diagup\diagdown}$ at the top. If the query has no H^1, V^1 or I^1, we search the corpus for a stripped D-tree matching the stripped Q-tree. If the query has H^1, V^1, or I^1, we extend the corpus by applying rules TQ6a, TQ6b, TQ6c, and then search the extended corpus. In extending the corpus, we apply only the one of the tree rules indicated by the Q-tree; we apply the rule by identifying first the pair of $\overset{Sh}{\diagup\diagdown}$-trees indicated in the Q-tree. We then trace up from those terminal nodes of a candidate tree in the corpus, simultaneously from both terminal points, and check whether the vertex of the path is as prescribed in the Q-tree.

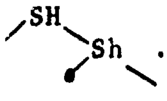
In comparing a stripped D-tree of the extended corpus with the stripped Q-tree, we also begin simultaneously at all terminal points and trace up the paths. We call a mismatch as soon as one of the vertices other than one which had a question-mark next to it, and proceed to another D-tree of the corpus. We do not examine trees <u>to</u> which rules TQ6 were applied, only the trees that result.

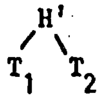
The D-trees and the trees corresponding to parsed pictures are in 1-1 correspondence. We have seen that many D-trees correspond to a given picture. We will call these equivalent. If a given picture corresponds to the answer to a query, there will again be many queries which correspond with the same answer. We will call these queries equivalent. We now seek a representative, a canonical member, of each of these two equivalence classes so that only one comparison between the canonical query and a canonical D-tree is needed. This will minimize the number of D-trees in the corpus that have to be compared.

V. OPTIMAL TREES FOR STORAGE AND SEARCH

The criterion for choosing the canonical representative of an equivalence class of trees in minimization of the expected number of elementary node-deletions necessary to transform a D-tree into the tree is specified by H', V', or I' in a Q-tree. To define an elementary node-deletion, consider an algorithm for applying rules TQ6. Suppose that the given Q-tree is 

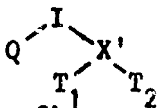
where T_1 and T_2 are two terminal nodes corresponding to, say, 

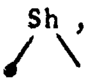
and 

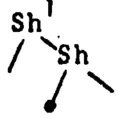
. Suppose that the following is a path in a D-tree with the same vertex and terminal nodes. To transform this path into 

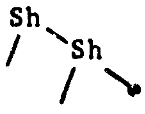
, we start with T_1 or T_2, whichever is lower in the D-tree. We trace to the node on the next level up and delete it, proceeding in this way until we are at the level of T_2 or T_1 whichever was higher in the D-tree. We now move up to the next-level node on both sides of the path, and check whether the paths intersect. If not, we delete both nodes, move up to the next level and repeat. If so, we check that this top-most vertex is H (as prescribed in the Q-tree) and terminate the process. (If it is not H we substitute what it is and place N next to it information the answer-tree, as indicated before .)

To make clear what we mean by the level of a node in a tree, we call the top-most node in the tree level 0. Thus, in Fig. 5.1, at level 0 we have H, at level 1, V and H, etc. Both T_1 and T_2 happen to be at level 8 in that example.

<u>Theorem 5.1</u>: Consider a query tree of the form 

, where X' stands for H', V' or I' and T_1, T_2 for 

, 

or 

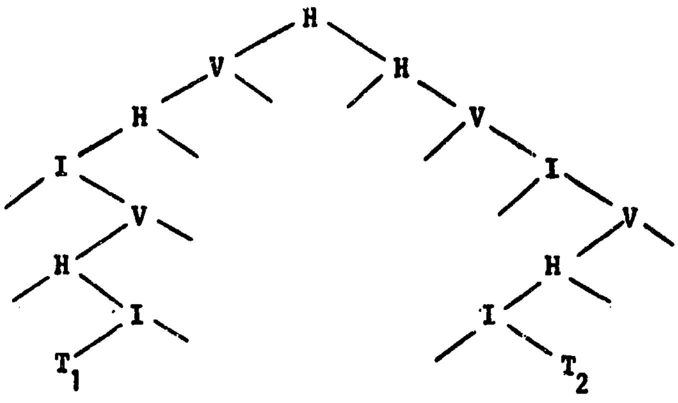
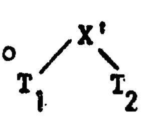


Fig. 5.1

Let $p(u,v,w)$ be the probability of T_1 being at level u, T_2 being at level v and the vertex at which the path up from T_1 intersects the path from T_2 being at level w. The expected number of elementary node-deletions to transform a circuit in a D-tree into  is

$$\sum_{x=1}^{d}\left[\left(\sum_{u}\sum_{v=u+1}^{d}+\sum_{v}\sum_{u=v+1}\right)p(u,v,v-x-1)\sum_{w}p(u,v,w)+\sum_{v}p(v,v,v-x-1)\sum_{w}p(v,v,w)\right].$$

Proof: Let L_2; L_1 denote the levels of T_2 and T_1 and suppose $L_2 > L_1$. It will take $L_2 - L_1 - 1$ deletions to get to the same level. It will take another L_1-L-1 deletions to reach the top vertex of the path on the shorter arm, L_2-L on the longer arm. Altogether, this is $2L_2$-2L-2 deletions. If $L_2 < L_1$, it will take $2L_1$-2L-2 deletions. The probability of L_2-L-1 being x is a convolution,

$$\sum_{L_2,L_1} P(L_2=v,\ l=v-x-1)P(L_2 > L_1)=\sum_{u=0}^{d}\sum_{v=u+1}^{d} P(L_2=V,L=V-X-1)P(L_2=V,\ L_1=u).$$

Here d is the lowest level of the D-tree, called its depth. Similarly the probability of L_1-L-1 being x is $\sum_{v=1}^{d} \sum_{u=v+1}^{d} P(L_1=u,\ 1=u-x-1)P(L_1=u,\ L_2 = v)$.

We can define $q(u,v) \equiv \sum_{w} p(u,v,w) = P(L_1=u,\ L_2 = v)$. The total probability of having x deletions is now

$$\sum_{u} \sum_{v=u+1}^{d} p(u,v,v-x-1)q(u,v) + \sum_{v} \sum_{u=v+1}^{d} p(u,v,v-x-1)q(u,v) + \sum_{v} p(v,v,v-x-1)q(v,v).$$

The expected number of deletions is the sum of this expression over x from 1 to d.

QED.

If p(u,v,w) is a uniform distribution, the expected number of node deletions will be proportional to d, the depth of D-trees. For distributions with a smaller variance than the uniform, the expected number may grow less slowly than d but never faster. Hence, minimizing d will minimize the upper bound on the expected number of deletions. Thus, if there is a number of equivalent D-trees that correspond to the same picture, use as a canonical representative of this equivalence class of D-trees the one with smallest d.

To illustrate, the two D-trees shown in Figures 5.2 and 5.3 are equivalent representations of the picture shown in Fig. 5.4.

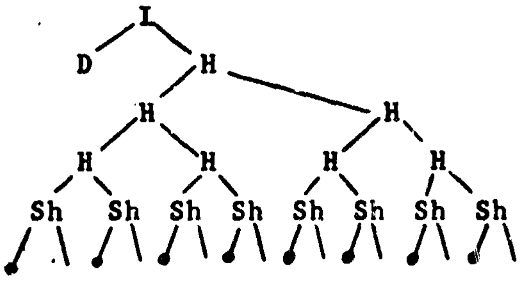


Fig. 5.2

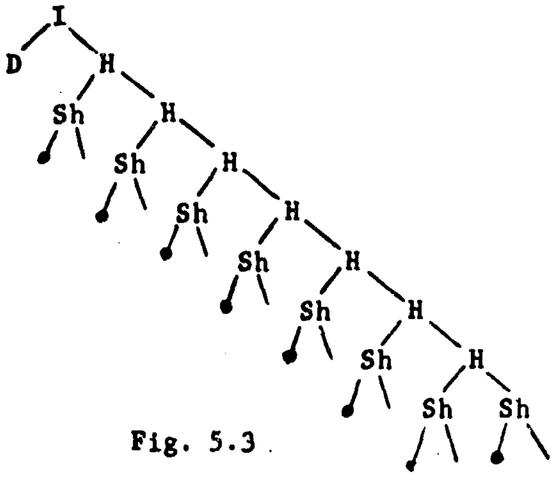


Fig. 5.3

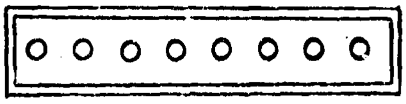


Fig. 5.4

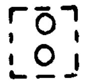
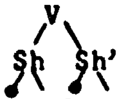
For the tree in 5.2, d=4, for the one in 5.3, d=8. Generally, d cannot be less than $\log_2 n$, where n is the number of objects or terminal nodes.

One of the shortcomings of a tree-query-language like L'_{TQ} is that it does not allow us to ask a question like $\begin{smallmatrix} & H' & \\ / & & \backslash \\ \underline{1} & & \underline{8} \end{smallmatrix}$, where $\underline{1}$ and $\underline{8}$ denote the first and 8th circle in Fig. 5.4. To ask such queries we must introduce naming which leads into the topic of name-languages to be treated in Section 6. Note that if we could ask this query, transforming the tree Fig. 5.2 into $\begin{smallmatrix} & H' & \\ / & & \backslash \\ \underline{1} & & \underline{8} \end{smallmatrix}$ would require 4 node-deletions; transforming tree (5.3) into it would require 6 node-deletions. Thus, 5.2 should be used as the standard representative of the picture of Fig. 5.4. A corpus of pictures to be searched for answers to queries will henceforth be stored in terms of such corresponding standard D-trees. Some of the details of how to store such trees in a computer memory, how to index a corpus of such stored trees, the programs for search — all aimed at efficiency — are given in Scient. Rpt. No. 3.

Trees and pictures are not at the same level. If a tree is used to describe the same data described by a picture, the tree corresponds to the <u>parsed</u> picture. A parsed tree corresponds to a parsing of a parsed picture. We can think of a tree-representation of data as further removed from an iconic representation of the data than is the pictorial representation. The English-like, symbolic language to be studied next is even further removed than the tree-language. The loss of iconic resemblance between the representational symbols and their designata is compensated by increased power of generalization, expressibility, and inference.

VI. ENGLISH-LIKE LANGUAGES

A. An English-Like Language for Describing "Pictorial" Data

A description of Fig. 6.1* in English words might read as follows: "Fig. 6.1 is a picture which consists of a square, two circles, and a triangle in vertical alignment. The square is at the top, the triangle at the bottom." The basic complete unit in this language, corresponding to a picture or a D-tree, is a paragraph, such as the above. Corresponding to "configuration," such as  , or a tree such as , is the sentence. This is a unit of the non-terminal vocabulary. Spaces and selected English words constitute the terminal vocabulary.

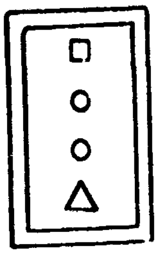


Fig. 6.1.

In what follows, we will confine our attention to paragraphs in standard but English-like form. Elsewhere we shall provide rules for transforming less constrained paragraphs, such as the one illustrated above, into these standard forms. The standard — form paragraph will consist of a single sentence. We will denote this form by DPGSENT. (MNEMONIC: Descriptive paragraph sentence). It is the unit U of the linguistic system $S(L_{ND})$; it is the distinguished element of the non-terminal vocabulary. Important other members of the non-terminal vocabulary are PICT, SPEC, NREL, SH, etc.

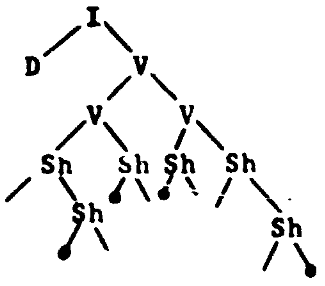


Fig. 6.2

[*] The description of Fig. 6.1 in tree language is shown in Fig. 6.2.

All elements of the non-terminal vocabulary will here be written in capitals; those of the terminal vocabulary in lower-case letters, except for the proper names of individual objects which begin with a capital letter and are underlined.

The rules of $S(L_{ND})$ are:

ND1: DPGSENT → NI + PICT

The right-hand side of this rule specifies a concatenation of two units, with a space between them, unit NI being to the left of unit PICT. These units, in turn, are specified by similar rules. Such rules are applied repeatedly until a string of words in the terminal vocabulary results.

ND15: NI → One, Two, Three ... (Mnemonic for NI: "Name" of Individual)
→ 1 , 2 , 3 , ...

The commas in this and similar rules denote "or". The terms on the right-hand side are elements of the terminal vocabulary; any one of them could be substituted for NI in ND15, and again in ND1. The three dots in ND15 are to suggest that any word "like" the first three — i.e., any English word that begins with a capital letter and is underlined — can also be substituted for NI.

ND2: PICT → IS + SPEC

ND3: SPEC → A + SP

ND4: SP → PCT + CSOF

ND5: CS OF → W + COF

ND6 a) COF → CONS + OBJS,
b) → AND + OBJS

ND7: OBJS → NI + CLS

ND8: CLS → W + CL

ND9: CL → IS + PROP

ND10: PROP → SH + REL

ND11: REL → AND + RELN

ND12: a) RELN → NREL + CONT
b) RELN → the last object.

ND13:	a)	CONT	→ NI + REL
	b)		→ NI + SC
ND14:		SC	→ ; + COF
ND16:		IS	→ is
ND17:		A	→ a, an
ND18:		PCT	→ picture
ND19:		W	→ which
ND20:		CONS	→ consists of:
ND21:	a)	SH	→ circular,
	b)		→ square,
	c)		→ triangular
ND22:		AND	→ and
ND23:	a)	NREL	→ below,
	b)		→ to the right of
	c)		→ to the left of
	d)		→ enclosing
	e)		→ above
	f)		→ inside

In applying a rule like 23b, the phrase "to the right of" is treated like a single word, as if the 3 spaces were not present. To describe Fig. 6.1, we have:

> "Figure 6.1 is a picture which consists of: One which is square and above Two; and Two which is circular and above Three; and Three which is circular and above Four; and Four which is triangular and below Three and the last object."

There are 41 words in this sentence (counting "consists of" and "the last object" as single words and the semicolons as words) and these are identified as follows:

Sentence words:	Figure 6.1	is	a	picture	which	consists of	
Corresponding non-terminal symbols:	NI	IS	A	PCT	W	CONS	etc.
Corresponding Rule:	ND15	ND16	ND17	ND18	ND19	ND20	

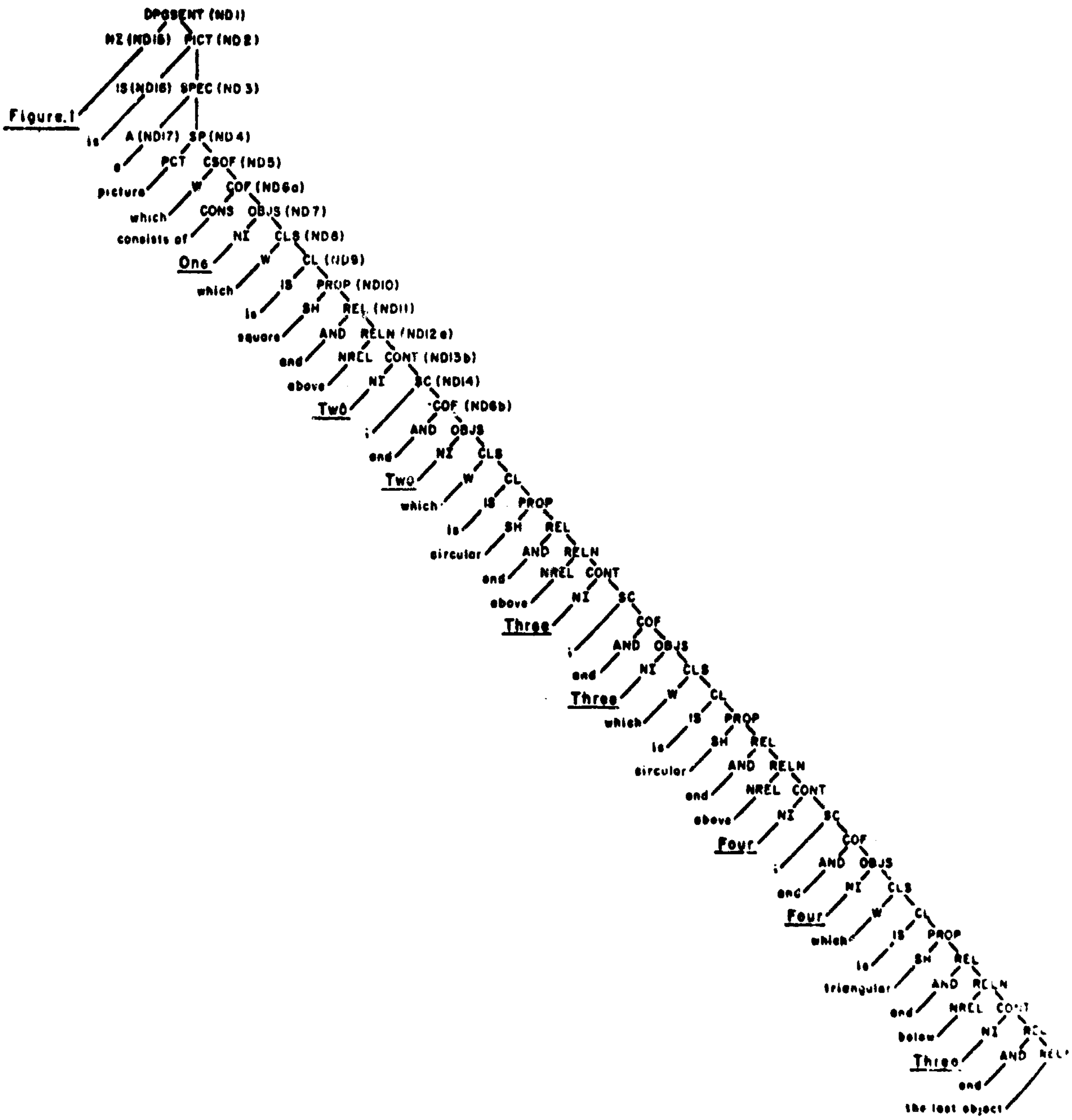

DPGSENT (ND1)
NI (ND15)
PICT (ND2)
IS (ND16)
SPEC (ND3)
Figure.1
is
A (ND17)
SP (ND4)
a
PCT
CSOF (ND5)
picture
W
COF (ND6a)
which
CONS
OBJS (ND7)
consists of
NI
CLS (ND8)
One
W
CL (ND9)
which
IS
PROP (ND10)
is
SH
REL (ND11)
square
AND
RELN (ND12a)
and
NREL
CONT (ND13b)
above
NI
SC (ND14)
Two
;
COF (ND6b)
AND
OBJS
and
NI
CLS
Two
W
CL
which
IS
PROP
is
SH
REL
circular
AND
RELN
and
NREL
CONT
above
NI
SC
Three
;
COF
AND
OBJS
and
NI
CLS
Three
W
CL
which
IS
PROP
is
SH
REL
circular
AND
RELN
and
NREL
CONT
above
NI
SC
Four
;
COF
AND
OBJS
and
NI
CLS
Four
W
CL
which
IS
PROP
is
SH
REL
triangular
AND
RELN
and
NREL
CONT
below
NI
REL
Three
AND
REL
and
the last object

Fig. 6.3

The language defined by these vocabularies and rules will lead to some "sentences" which are unwanted; for example, nothing makes us discard a sentence having in it the clause "above *Two* and below *Two*," though this is evidently contradictory. It will not generate very ungrammatical sentences. Many grammatically better and equivalent statements are missed. We do, however, claim:

Theorem 6.1: The name-language L_{ND} is equivalent to L_{GD}.

Proof: In identifying items in the terminal vocabulary of $S(L_{GD})$ with items on the terminal vocabulary of $S(L_{TD})$ we must identify two levels of names: proper names of individual object tokens and generic names of object-types. (This is what we did not try to do in non-name languages.) Identify $\boxed{\square}$ with a proper name like *Figure 6.1* and with the word "picture" in the form "*Figure 6.1* is a picture which consists of:" Identify ○ with a proper name as well as with "circular"; etc.

Given any picture in L_{GD}, we always begin by forming the phrase "*Name* is a picture which consists of:", according to rules ND15, 16, 17, 18, 19, 20. Here *Name* stands for an arbitrary name we assigned to the picture. We now assign different names to all the objects in the picture. We then form a clause for each object, which always starts with "*Name* which is (Shape) and ⋯." In place of shape we insert circular, triangular or square. We must form as many such clauses as we have different names. We complete the clause — fill in ⋯ — by "to the right of *Name* and to the left of *Name* and above *Name* and below *Name* and inside *Name* and (the last object.)" however many of these apply. We excluded mention of "enclosing Name", for that information is picked up when the latter name appears at the head of a clause. Otherwise, however, we do not try to eliminate redundant information. This procedure will: a) form a well-formed sentence according to rules ND1 - ND23; b) describe the given picture.

We can construct an equivalent picture from any sentence in L_{ND} by making the appropriate identifications.

To say that L_{GD} and L_{ND} are equivalent is to say that we can transform each "sentence" of the other language, not that there is a 1-1 correspondence between the two languages. Indeed, to a given picture correspond many sentences of L_{ND} and they are informationally equivalent to each other. Conversely, to each sentence of L_{ND} correspond a number of equivalent pictures which differ in metric and other respects we have here ignored.

By the parsing tree associated with a sentence of L_{NG} we mean the tree obtained as a result of applying various rules to the sentence. The tree has at its nodes the vocabulary items and rule names. As an example consider the parsing tree for the sentence stated at the beginning of this section — it is shown in Fig. 6.3.

B. The English-Like Language for Specifying Constructions

We wish to construct a language of imperative sentences such that executing the directives results in sentences of L_{ND}. We shall use the word "order" to designate the units of L_{NC}. In constructing $S(L_{NC})$ it is helpful to think of the pictures associated with the sentence of L_{ND} which is generated on execution of an order. In this picture each object, each configuration and the picture itself is assumed to be named. A typical order is: "Name 1 is an order specifying: configuration 1 enclosing configuration 2 above configuration 3; configuration 2 enclosing Name 2 which is square and enclosing configuration 4; configuration 3 enclosing configuration 5 to the left of configuration 6; configuration 4 enclosing Name 3 which is circular; configuration 5 enclosing Name 4 which is circular and encloses configuration 7; configuration 6 enclosing Name 5 which is triangular; configuration 7 enclosing Name 6 which is circular."

We will specify rules of construction, like C1-C6c in Section III which take the above statement into a pictorial construction statement in L_{GC}. We must first develop rules for forming statements like the above. These are:

NC1		CPGSENT	→ NI + ORDER
NC2		ORDER	→ IS + A + ORDSP
NC3		ORDSP	→ ORD + SPING + SPEC
NC4		SPEC	→ VAR + ENCL
NC5		VAR	→ CONF + NUM
NC6		ENCL	→ ENC + PR
NC7	a)	PR	→ VAR + NREL + VAR + SC
	b)		→ NI + CL
NC8		CL	→ W + IS + REST
NC9	a)	REST	→ SH + SC
	b)		→ SH + AND + EN
	c)		→ SH + PER
NC10		EN	→ ENC + VAR + SC
NC11		SC	→ SHICOL + SPEC
NC12		ORD	→ order

NC13	CONF	→ configuration
NC14	NUM	→ 1, 2, 3, ···
NC15	ENC	→ enclosing
NC16	SPING	→ specifying
NC17	IS	→ is
NC18	A	→ a, an
NC19	W	→ which
NC20	PER	→ .
NC21	SHICOL	→ ;

Rules for Sh and NREL are the same as rules ND21, ND23; NI as in ND15.

There is a last important rule which cannot be expressed in the form of a production. It is NC22: The numerals following every occurrence of the word "configuration" must be assigned so that each string of terminal element between semicolons begins with "configuration i..." in the order i = 1, 2, 3, ..., n. No numeral larger than n can appear anywhere. Each numeral except 1 must occur exactly twice.

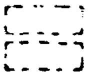
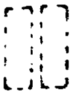
Given a statement like the one illustrated above, we parse it first: that is, we form a labelled bracketing or, equivalently, a parsing tree, which is shown in Fig. 6.4.

The figure, for brevity, covered only the first two clauses of the specification in the order. At each node of this tree should also be the name of rules used to obtain the non-terminal element at that node. Thus, the top node is the left-hand side of rule NC1.

To construct a pictorial order, proceed as follows, using the parsing tree illustrated in Fig. 6.4.

CN1 Examine the top node. If it is CPGSENT (NC1), draw a wavy line frame.

CN2 Scan the tree from the top down to the first occurrence of SPEC(NC4). When located, draw a dotted line square inside the wavy line frame. To determine the order in which this construction proceeds, trace from SPEC to the nearest NUM; it should, in this step, be 1.

CN3 Trace down the tree from SPEC to FNCL(NC6). Trace one step down to locate PR. If Rule (NC7a) applies, draw  or  inside the dotted line square just completed depending on whether rule (ND23e) or (ND23c) applies.

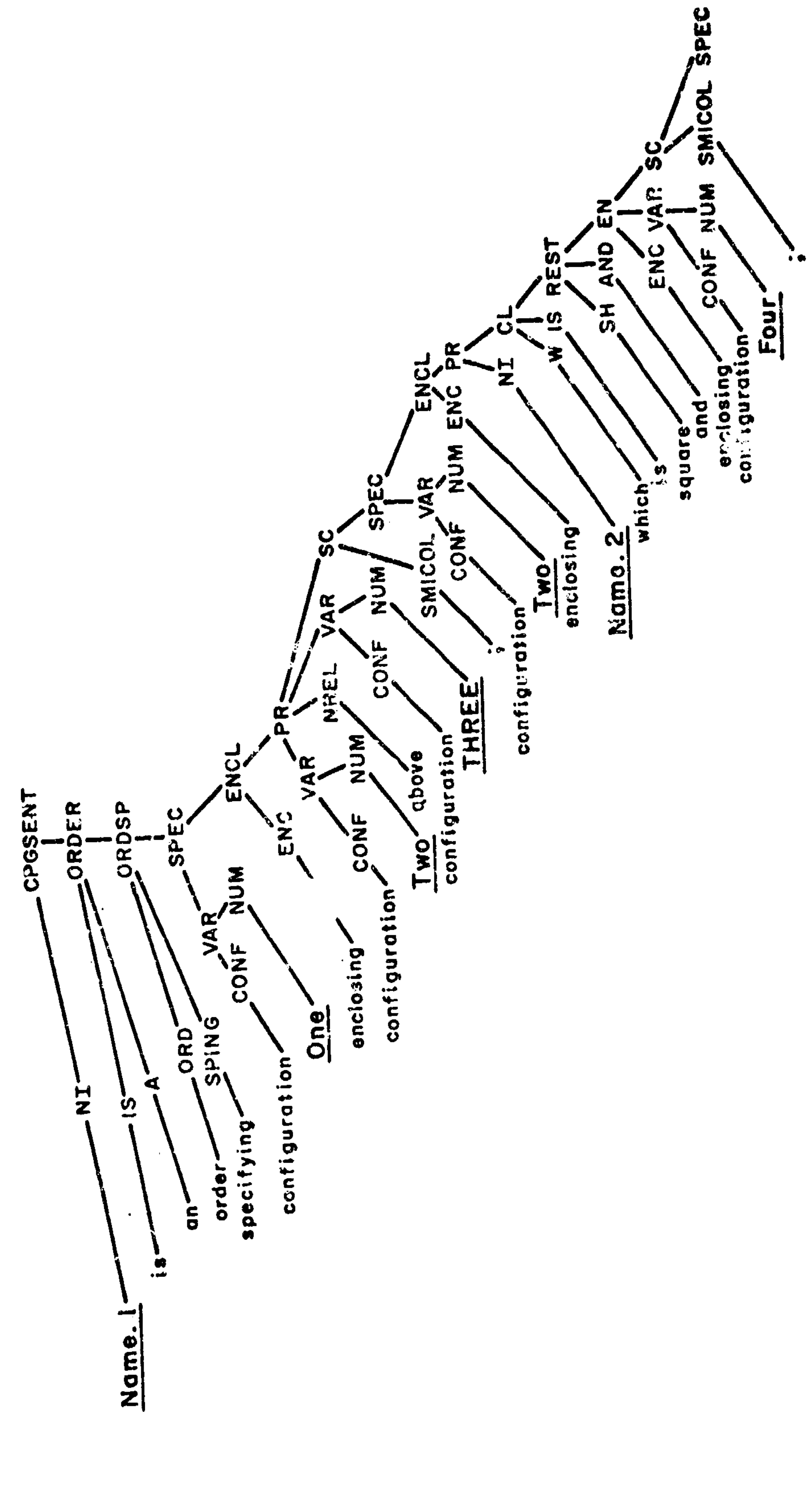

CPGSENT
NI
Name. 1
ORDER
IS
is
A
an
ORDSP
ORD
order
SPING
specifying
configuration
SPEC
VAR
CONF
configuration
NUM
One
enclosing
configuration
ENCL
ENC
PR
VAR
CONF
configuration
NUM
Two
NREL
above
VAR
CONF
configuration
NUM
THREE
SC
SMICOL
;
SPEC
VAR
CONF
configuration
NUM
Two
ENCL
ENC
enclosing
PR
NI
Name. 2
CL
W
which
IS
is
REST
SH
square
AND
and
EN
ENC
enclosing
VAR
CONF
configuration
NUM
Four
SC
SMICOL
;
SPEC

Fig. 6.4

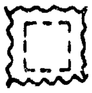
CN4 If Rule (NC7b) applies at PR, trace down to REST. If Rule NC9a applies at REST, draw ○ , □ , or △ , inside the dotted-line square just drawn, depending on whether (ND21a), ND21b) or (ND21c) applies at Sh.

CN5 If Rule (NC9b) applies at REST, draw , , inside the square just drawn depending on whether (ND21a, b, or c) applies at Sh.

CN6 If Rule (NC9c) applies at REST, the construction is completed.

<u>Theorem 6.2</u>: To each construction order in L_{NC} corresponds a unique pictorial construction specification in L_{GC}.

<u>Proof</u>: Every element of L_{NC} has a parsing tree with nodes labeled CPGSENT, SPEC, ENCL, REST, SH by rules of $S(L_{NC})$. Hence the algorithm CN1-CN6 applies to each statement of L_{NC}. There is only one way of tracing down the parsing tree, so that the nodes specified in CN1-CN6 are reached in a unique order if we parse the sentence of L_{NC} in a particular way (e.g., from left to right).

We must show that the result of applying steps CN1-CN6 is always an element of L_{GC}. To show this note that steps CN1 and CN2 together produce , which is the results of applying both rules (1') and (4) of $S(L_{GC})$. Rule CN3 correspond to rules (2a), (2b) plus rule (4) of $S(L_{GC})$. Rule CN4 corresponds to rules (3a),(3b),(3c) plus rule 4. Rule CN5 corresponds to rules (2c),(2d),(2e) plus rule (4). Rule 6 insures that the conversion process from L_{NC} to L_{GC} terminates. Figure 6.5 illustrates the pictorial construction specified by the statement examplified here. The numbers attached to the dotted-line squares indicate the order in which they were drawn. Names can be omitted.

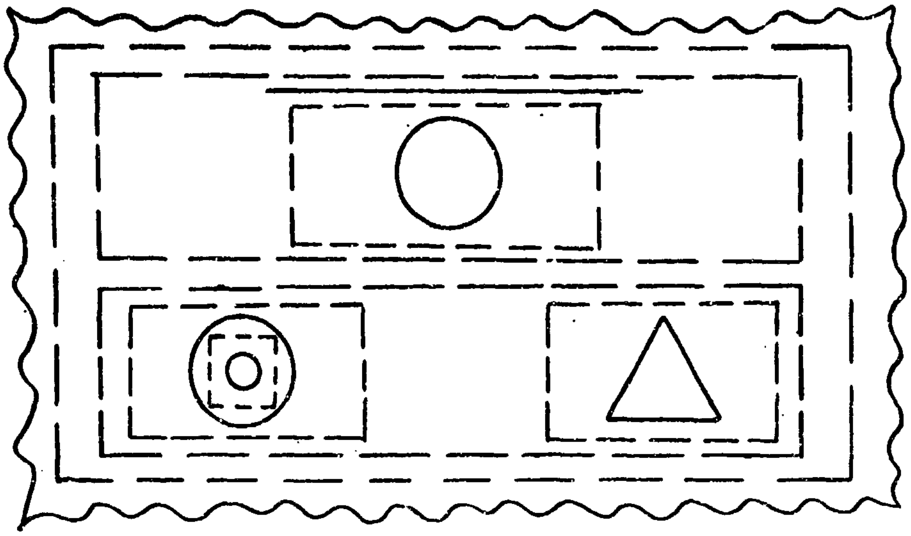


Fig. 6.5

Theorem 6.3: To each order in L_{NC} corresponds a descriptive statement in L_{ND}.

Proof: Construct a pictorial construction specification, the possibility of which is guaranteed by theorem 6.2. Then execute it, according to rules C1-C6e, to form a picture in L_{GD}. Then proceed to describe that as outlined in the proof of theorem 6.1. The result is a statement in L_{ND} as proved in theorem 6.1.

Theorem 6.4: To each order in L_{NC} corresponds a construction tree in L_{TC}.
Proof: Theorem 6.2 in Section VI, asserts that L_{GC} and T_{TC} are in one-one correspondence. Hence, by the above theorem 6.2, the result follows. To illustrate, Fig. 6.6 shows the tree corresponding to Fig. 6.5. From this we can immediately get the corresponding D-tree.

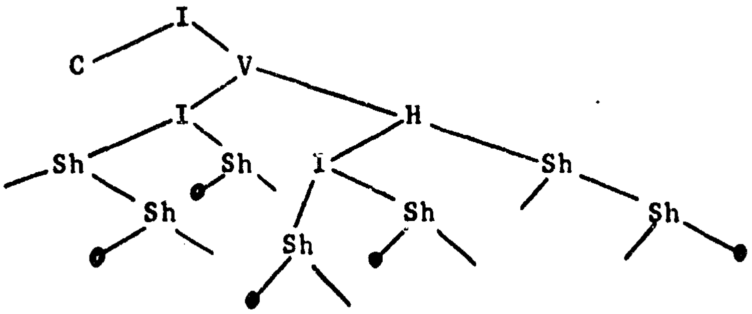


Fig. 6.6

C. The Name Language Specialized for Posing Queries and Stating Answers

The "simplest" queries involve verification of a specified statement. This is similar to a query about the truth or falsity of a proposition. Thus, we can obtain query sentences from orders simply by replacing the beginning of an order, "Name 1 is an order specifying: ..." by "is it true that Name 1 consists of: ...". The remainder of the order is unchanged.

Actual queries will never specify the entire context such as all the seven clauses in the example of Section 6.2. That is why the entire paragraph-sentence has a name, Name 1. Special parts of a clause may be designated for verification. A typical query might be: "In Name 1 is it true that: Name 6 is to the left of Name 5." With a slight variation of the beginning we can get: "In Name 1 find ? such that: Name 6 is to the left of ?".

Similarly, the answer to such a query need not produce unwanted (e.g., irrelevant to the query) statements of Name 1 It could be a simple "Yes, in Name 1 it is true that: Name 6 is to the left of Name 5", or "In Name 1, Name 5 is such that Name 6 is to the left of it".

In this section we try merely to relate L_{NQ} and L_{NA} to the corresponding sublanguages in L_T and L_G. We will elsewhere develop a more general query language together with algorithms to translate linguistic queries of a deeper sort directly into efficient tree-searching programs.

Consider, first, rules for a system $S_1(L_{NQ})$, which are:

N_1Q1: QPGSENT → QPRE + SPEC.

N_1Q2: QPRE → QTR + NI + CONS

N_1Q3: QTR → is it true that

The remaining rules are ones introduced earlier, namely

ND20: CONS → consists of:

ND15 for NI, NC4-22 for SPEC.

We can construct a pictorial query from a parsed query sentence in this language by proceeding as in CNI-CNG, plus inserting ? into each □, ○, and △, and instead of drawing a wavy-line frame as stated in CNI we draw a curly-line frame. This proves:

Theorem 6.5: To every query constructed according to $S_1(L_{NQ})$ corresponds a pictorial answer in L_{GA}.

This answer will be a frame with curly-lines with a curly-line frame around each object.

By an answer in the system $S_1(L_{NA})$ we mean a sentence of the form: "It is true that Name 1 consists of: ..".

N_1A1: APGSENT → APRE + SPEC

N_1A2: APRE → ATR + NI + CONS

N_1A3: ATR → It is true that

The remaining rules are as in $S_1(L_{NQ})$. To construct a pictorial answer from any such answer-sentence, draw a curly-line frame around each □, ○ and △. By parsing a pictorial answer, then removing the curly-line frames and applying rules N_1A1, N_1A2, in reverse, we can construct an answer-sentence in L_{NA}. Thus we have:

Theorem 6.7: For every query constructed according to $S_1(L_{NG})$, there is an appropriate answer constructed according to $S_1(L_{NA})$.

System $S_1(L_{NQ})$ is limited. The rules of $S_2(L_{NQ})$ are:

N_2Q1: QSENI → QINTR + QSPEC

N_2Q2: QINTR → IN + NI + QTR2

N_2Q3: IN → in

N_2Q4: QTR2 → is it true that:

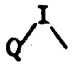
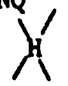
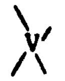
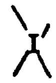
N_2Q5: QAPEC → NI + SMPR + NI

N_2Q6: SMPR → IS + SM + NREL

N_2Q7: SM → somewhere

The rules of NI, IS, NREL are as stated before. The sentence "In Name 1 is it true that: Name.2 is somewhere to the right of Name.3" is a typical product of these rules. We have not tried to enrich this query language by even allowing questions about shape; we wish merely to relate this query language to the language $\acute{L}_{TQ}$ developed in Section IV. To do this, we replace the trees beginning with Sh , as the terminal nodes of a Q-tree, by proper names.

Theorem 6.8: To each query-sentence formed according to $S_2(L_{NQ})$ corresponds a query-tree in L_{GP}.

Proof: Suppose that the parsing tree of a sentence in L_{NQ} is given. At node QSENT(N_2Q1), form . At node SMPR(N_2Q6) attach , ,  to the Q-graph, depending on which of the rules of ND23 are applied at node NREL. Attach the names specified at node QSPEC(N_2Q5) in proper order to complete the Q-tree. We will also attach the name of the figure to be search next to Q on the Q-tree, when we reach node QINTR after applying rule N_2Q2 in the parsing tree. The result is a tree with the two mentioned modifications, and thus as a tree of the extended tree-query language L_{TQ} defined previously (Section IV).

Theorem 6.9: To each query sentence formed according to $S_2(L_{NQ})$ there is a pictorial answer in L_{GP}.

Proof: First form the Q-tree in L_{TQ} according to the preceding theorem. Next, process the Q-tree according to the algorithm of Section IV, search the corpus specified by the name next to Q. In the present extension of our various means of representation, we suppose that each corpus that can be searched separately is given a name, and that name is recorded with it. Similarly, all objects are named and names are recorded with them. In testing for match, the recorded names must coincide with names specified in the query. From the matching trees in the corpus, pictorial answers may be formed by the procedure indicated TQS1-TQS4.

We can now construct an system $S_2(L_{NA})$ analogously to $S_1(L_{NA})$ and show:

Theorem 6.10: For every query constructed according to $S_2(L_{NQ})$, there is an appropriate answer constructed according to $S_2(L_{NA})$.

Proof: The rules of $S_2(L_{NA})$ are:

N_2A1: ASENT → AINTR + ASPEC

N_2A2: AINTR → IN + NI + ATR2

N_2A3: ATR2 → it is true that:

all other rules are as in $S_2(L_{NQ})$.

We form a sentence of L_{NA} according to these rules from a pictorial answer by parsing the pictorial answer, removing the curly-line frames and applying N_2A1, N_2A2, N_2A3, etc., in reverse. Thus, given a query, we form the corresponding Q-tree, process it to produce a pictorial answer, then describe the latter as a sentence in L_{NA}.

We conclude by introducing queries with question marks. Consider the system $S_3(L_{NQ})$:

N_3Q1: QUERY → QNMFD + QVSPEC

N_3Q2: QNMFD → IN + NI + FDST

N_3Q3: FIND → find ? such that:

N_3Q4: QVSPEC → VI + SMPR + VI

N_3Q4: VI → NI , ?

All other rules, for SMPR, are as before. The main difference is that we can use ? in place of proper names. We transform such queries into Q-trees modified in that names are attached to the terminal nodes and to the Q-node. We then proceed as we did for $S_2(L_{NQ})$ to produce pictorial answers. We construct answer-sentences from such pictorial answers according to rules of $S_3(L_{NA})$.

N_3A1: AVERY → ANMFD + AVSPEC

N_3A2: ANMFD → IN + NI + NMST

N_3A3: NMST → COM + NIST

N_3A4: NIST → NI + IS + ST

N_3A5: ST → such that

N_3A6: COM → ,

N_3A7: AVSPEC → AI + SMPR + AI

N_3A8: AI → NI, IT

N_3A9: IT → it.

The rules for NI, SMPR, IS, IN, etc., are as before. In forming an answer in L_{NA}, these rules are used in reverse. The rule AI → IT is used to place "it" where a question mark appeared in the query, and where the name which is now introduced in rule N_3A4 appeared in a matching picture.

The sentence: "In Name.1, Name 5 is such that Name.2 is to the left of it" is in L_{NA}. It corresponds to the query "In Name.1 find ? such that: Name.2 is to the left of ?".

Theorem 6.11: To every query constructed according to $S_3(L_{NQ})$, there is an appropriate answer constructed according to $S_3(L_{NA})$.

VII. CONCLUDING COMMENTS

We have shown how to construct descriptive, constructive, interrogative and responsive languages in graphic, tree and English-like representations for a very simple domain of discourse. We have shown how to connect these various sub-languages. We have seen that a tree representation has advantages over the other two means of representation for automated storage and search of the kind of data considered; that an English-like representation has advantages for posing queries; that a graphic representation has advantages for displaying answers.

We have not yet shown how far we can extend the English-like language toward ordinary English to pose a greater variety of queries of the same data; nor have we as yet shown how to translate directly from English-like queries into efficient computer search programs. Some of our work in this area will be presented in a forthcoming paper. Also, we have not paid any attention to the important problem of how to extend these ideas to more complex, more varied, more realistic types of data, and how to automatically form the language system as the data base expands. These questions are currently under study.

REFERENCES

1. Amarel, S., "On the Automatic Formation of a Program which Represents a Theory," in Self-Organizing Systems-62, Ed. Yovits Jacobi and Goldstein, Spartan Books, 1962, pp. 107-175.

2. Bohnert, H. G., "English-Like Systems of Mathematical Logic for Content Retrieval," Automation and Sci. Communic., H. P. Lohn, Ed., ADI 26th Annual Meeting, Oct. 1963, p. 155.

3. Climenson, W. D., Hardwick, N. H., and Jacobson, S. N., Intelligence Syntax Analysis, RCA Report, December 31, 1962.

4. Cooper, W. S., "Fact-Retrieval and Deductive Question-Answering Information Retrieval Systems," J. of ACM 11, 2, April 1964, pp. 117-137.

5. Darlington, T. L., "Translating Ordinary Language Into Symbolic Logic," MAC-M-149, MIT, Cambridge, Mass., March 1964.

6. McNaughton, R., On Computations Executed by Multi-Computer Systems, RCA Report, August 16, 1963.

7. Kirsch, R., "Computer Interpretation of English Text and Picture Patterns," IEEE Trans. EC-13, August 1964, No. 4, p. 363.

8. Simmons, R. and Londe, D., "Namer: A Pattern Recognition System for Generating Sentences About Relations Between Line Drawings," Proc. National ACM Conf., August 24-26, 1965, pp. 162-175.

9. Sutherland, I., "Sketchpad, A Man-Machine Graphical Communication System," Proc. SJCC 23, 1963, pp. 319-326.

10. Minsky, M., "Steps Toward Artificial Intelligence." Proc. IRE, January 1961, p. 8.

UNCLASSIFIED
Security Classification

DOCUMENT CONTROL DATA - R&D

(Security classification of title, body of abstract and indexing annotation must be entered when the overall report is classified)

1 ORIGINATING ACTIVITY (Corporate author)

RCA Laboratories
Princeton, New Jersey 08540

2a REPORT SECURITY CLASSIFICATION
Unclassified

2b GROUP
N/A

3 REPORT TITLE

ON THE REPRESENTATION OF LIMITED INFORMATION BY MEANS OF PICTURES, TREES, AND ENGLISH-LIKE SENTENCES

4 DESCRIPTIVE NOTES (Type of report and inclusive dates)

SCIENTIFIC; INTERIM

5 AU

Kochen, Manfred

6 REPORT DATE May 1968	7a TOTAL NO OF PAGES 60	7b NO OF REFS 10

8a CONTRACT OR GRANT NO
AF 49(638)-1184

b PROJECT NO [illegible]

c [illegible]

d [illegible]

9a ORIGINATOR'S REPORT NUMBER(S)

9b OTHER REPORT NO(S) (Any other numbers that may be assigned this report)

10 AVAILABILITY LIMITATION NOTICES

1. This document has been approved for public release and sale; its distribution is unlimited.

11 SUPPLEMENTARY NOTES

[illegible]

12 SPONSORING MILITARY ACTIVITY

Air Force Office of Scientific Research
Office of Aerospace Research
Arlington, Va. 22209

13 ABSTRACT

In this paper we discuss means of representing states of the world which are easily described as pictures of triangles, circles, and squares in horizontal, vertical, or enclosure relationships; our study is oriented to the comparative evaluation of different representations for computer-based question-answering systems.

Three languages for representing such pictorial data are constructed. The basic units of the first are pictures, of the second trees, and of the third sentences. Each of the three languages is further modified to serve for describing data, for specifying constructions, for posing queries, and for stating answers. The interrelations among the various specialized uses of these three languages are investigated. Queries are best posed in an English-like language, computer search best proceeds on data represented as trees, and answers can often be best presented in picture representations. Results are in the form of a) context-free generative grammars for the different languages expressed as production rules, b) theorems showing correspondences between, say, all query sentences and all pictorial answers, and c) formula for the effort to search for answers, for optimal trees to store data.

UNCLASSIFIED
Security Classification

14 KEY WORDS	LINK A		LINK B		LINK C	
	ROLE	WT	ROLE	WT	ROLE	WT
Question-answering systems Graphic languages Computer linguistics Tree processing Context-free grammar						

INSTRUCTIONS

1. ORIGINATING ACTIVITY: Enter the name and address of the contractor, subcontractor, grantee, Department of Defense activity or other organization *(corporate author)* issuing the report.

2*a*. REPORT SECURITY CLASSIFICATION: Enter the overall security classification of the report. Indicate whether "Restricted Data" is included. Marking is to be in accordance with appropriate security regulations.

2*b*. GROUP: Automatic downgrading is specified in DoD Directive 5200.10 and Armed Forces Industrial Manual. Enter the group number. Also, when applicable, show that optional markings have been used for Group 3 and Group 4 as authorized.

3. REPORT TITLE: Enter the complete report title in all capital letters. Titles in all cases should be unclassified. If a meaningful title cannot be selected without classification, show title classification in all capitals in parenthesis immediately following the title.

4. DESCRIPTIVE NOTES: If appropriate, enter the type of report, e.g., interim, progress, summary, annual, or final. Give the inclusive dates when a specific reporting period is covered.

5. AUTHOR(S): Enter the name(s) of author(s) as shown on or in the report. Enter last name, first name, middle initial. If military, show rank and branch of service. The name of the principal author is an absolute minimum requirement.

6. REPORT DATE: Enter the date of the report as day, month, year; or month, year. If more than one date appears on the report, use date of publication.

7*a*. TOTAL NUMBER OF PAGES: The total page count should follow normal pagination procedures, i.e., enter the number of pages containing information.

7*b*. NUMBER OF REFERENCES: Enter the total number of references cited in the report.

8*a*. CONTRACT OR GRANT NUMBER: If appropriate, enter the applicable number of the contract or grant under which the report was written.

8*b*, 8*c*, & 8*d*. PROJECT NUMBER: Enter the appropriate military department identification, such as project number, subproject number, system numbers, task number, etc.

9*a*. ORIGINATOR'S REPORT NUMBER(S): Enter the official report number by which the document will be identified and controlled by the originating activity. This number must be unique to this report.

9*b*. OTHER REPORT NUMBER(S): If the report has been assigned any other report numbers *(either by the originator or by the sponsor)*, also enter this number(s).

10. AVAILABILITY/LIMITATION NOTICES: Enter any limitations on further dissemination of the report, other than those imposed by security classification, using standard statements such as:

(1) "Qualified requesters may obtain copies of this report from DDC."

(2) "Foreign announcement and dissemination of this report by DDC is not authorized."

(3) "U. S. Government agencies may obtain copies of this report directly from DDC. Other qualified DDC users shall request through
____."

(4) "U. S. military agencies may obtain copies of this report directly from DDC. Other qualified users shall request through
____."

(5) "All distribution of this report is controlled. Qualified DDC users shall request through
____."

If the report has been furnished to the Office of Technical Services, Department of Commerce, for sale to the public, indicate this fact and enter the price, if known.

11. SUPPLEMENTARY NOTES: Use for additional explanatory notes.

12. SPONSORING MILITARY ACTIVITY: Enter the name of the departmental project office or laboratory sponsoring *(paying for)* the research and development. Include address.

13. ABSTRACT: Enter an abstract giving a brief and factual summary of the document indicative of the report, even though it may also appear elsewhere in the body of the technical report. If additional space is required, a continuation sheet shall be attached.

It is highly desirable that the abstract of classified reports be unclassified. Each paragraph of the abstract shall end with an indication of the military security classification of the information in the paragraph, represented as *(TS)*, *(S)*, *(C)*, or *(U)*.

There is no limitation on the length of the abstract. However, the suggested length is from 150 to 225 words.

14. KEY WORDS: Key words are technically meaningful terms or short phrases that characterize a report and may be used as index entries for cataloging the report. Key words must be selected so that no security classification is required. Identifiers, such as equipment model designation, trade name, military project code name, geographic location, may be used as key words but will be followed by an indication of technical context. The assignment of links, rules, and weights is optional.